Universidad Carlos III de Madrid

Departamento de Ingeniería de Sistemas y Automática

Curso 2013-2014

Trabajo Fin de Grado

Diseño de aplicación basada en Kinect para vehículo inteligente

Autor:

César Moreno Pulido

Tutor:

Fernando García Fernández

Co-tutor:

Pablo Marín Plaza

Agradecimientos

Me gustaría agradecer de forma incondicional a mis padres la confianza que siempre han depositado en mí. Gracias por inculcarme los valores que me han enseñado desde pequeño y que me han ayudado a ser mejor persona. Sé que sin vosotros no hubiera llegado hasta aquí. Gracias, os quiero.

También me gustaría destacar a mi hermano, siempre es reconfortante saber que alguien en esta vida quiere seguir tu ejemplo y eso me ha hecho coger muchas fuerzas para seguir adelante en este camino. Como hermano y amigo, siempre te apoyaré en todo lo que necesites intentando darte siempre los mejores consejos. Te quiero.

En especial, quería agradecérselo a la persona que ha cambiado mi vida por completo, mi pareja. La conocí hace más de dos años y se ha convertido en mi principal sustento en esta vida. Gracias por estar a mi lado, sin tu confianza y apoyo no habría conseguido lograr mis sueños. Te quiero muchísimo mi vida.

Al resto de familiares, tíos, primos, abuelos y familiares de mi pareja, por creer en mí y por aportarme sabios consejos que día a día también me hacen crecer como persona.

Quisiera hacer referencia a mis tutores, Fernando y Pablo, los cuales me han guiado y aconsejado en la realización de este proyecto, habiéndome dado la oportunidad de aportar un pequeño grano de arena en el campo de la investigación.

Diseño de aplicación basada en Kinect para vehículo inteligente

ÍNDICE DE CONTENIDOS

Índice de contenidos

Capítulo 1: Introducción 13
1.1. Motivación del proyecto 14
1.2. Objetivo del proyecto 14
1.3. Estructura y resumen del proyecto 15
Capítulo 2: Estado del arte 17
2.1. La visión artificial y su evolución histórica 18
2.2. Visión artificial en la automoción 20
2.3. Modelos de análisis del entorno 22
2.4. Sistemas de detección facial 23
2.5. Modelado y tecnologías de adquisición de datos 3D 25
Capítulo 3: Software y hardware 29
3.1. ROS 30
3.1.1. Información general. ROS Hydro Medusa 31
3.1.2. Objetivos de ROS 31
3.1.3. Conceptos generales 32
3.1.4. *catkin* vs *rosbuild* 34
3.1.5. Librerías 35
3.1.5.1. Point Cloud Library 35
3.1.5.2. OpenCV 36
3.1.6. Lenguajes de programación. Lenguaje C++ 37
3.1.7. Entorno de programación. QtCreator 2.3 37
3.2. SENSOR KINECT 39
3.2.1. Características principales y especificaciones técnicas 39
3.2.2. Calibración 42
3.2.3. Kinect vs cámaras convencionales 45
Capítulo 4: Diseño y desarrollo del proyecto 47
4.1. Planteamiento del problema 48
4.2. Desarrollo del proyecto 49
4.2.1. Adquisición de la nube de puntos 49
4.2.1.1. Sincronización en tiempo real 50
4.2.1.2. Adquisición de datos de profundidad 50

4.2.1.3. Adquisición de datos de color 51
4.2.1.3.1. Modelo Pin-Hole 52
4.2.1.4. Correlación de datos de profundidad y color 54
4.2.2. Detección de la orientación de la cara 56
4.2.2.1. Detección facial y segmentación 57
4.2.2.2. Identificación de los ángulos de la cara 58
4.2.3. Visualización 60
4.3. Ejecución de la aplicación 62
Capítulo 5: Resultados prácticos 66
5.1. Análisis de los movimientos realizados 67
5.2. Parámetros iniciales 69
5.2.1. Iluminación 69
5.2.2. Posición y orientación del sensor 70
5.3. Análisis de los resultados 72
5.3.1. Movimiento Pitch 72
5.3.2. Movimiento Yaw 74
5.3.3. Movimiento Roll 76
Capítulo 6: Conclusiones 78
6.1. Conclusiones generales 79
6.2. Principales dificultades y posibles mejoras 80
6.3. Futuras aplicaciones del proyecto 84
Capítulo 7: Presupuesto 86
7.1. Información general 87
7.2. Coste de equipos y material 87
7.3. Coste de personal 88
7.4. Coste total del proyecto 89
Capítulo 8: Bibliografía 90
Bibliografía 91
Anexo I. Datos movimiento Pitch 99
Anexo II. Datos movimiento Yaw 104
Anexo III. Datos movimiento Roll 110
Anexo IV. Datos falso positivo 115

Diseño de aplicación basada en Kinect para vehículo inteligente

ÍNDICE DE FIGURAS

Índice de figuras

Figura 1. El ojo humano 18
Figura 2. Métodos de detección de bordes 19
Figura 3. Método de detección de esquinas 20
Figura 4. Cámara de tráfico 21
Figura 5. Radar fijo con cámara 21
Figura 6. Sistema de detección de punto ciego Nissan 22
Figura 7. Regiones rectangulares 24
Figura 8. Esquema de funcionamiento Algoritmo Viola-Jones 25
Figura 9. Vista frontal y lateral de una nube de puntos en color 26
Figura 10. Cámara de tiempo de vuelo 27
Figura 11. Sensor Kinect 28
Figura 12. Logo ROS (Robot Operating System) 30
Figura 13. Logo ROS Hydro Medusa 31
Figura 14. Funcionamiento de ROS 33
Figura 15. Logo Point Cloud Library 36
Figura 16. Logo OpenCV 36
Figura 17. Logo QtCreator 38
Figura 18. Entorno de programación QtCreator 2.3 38
Figura 19. Partes de sensor Kinect para XBOX 360 39
Figura 20. Transformador para Kinect 40
Figura 21. Patrón de puntos sobre una pared blanca 41
Figura 22. Distorsión del patrón al introducir un objeto en el entorno 42
Figura 23. Funcionamiento de Kinect 42
Figura 24. Herramienta *camera_calibration* 43
Figura 25. Tablero sin tapar y tapando el emisor infrarrojo, respectivamente 43
Figura 26. Movimientos del tablero 44
Figura 27. Herramienta *rqt_reconfigure* 44
Figura 28. Color y profundidad en tiempo real 50
Figura 29. Nube de puntos profundidad 51
Figura 30. Esquema de conversión ROS-OpenCV 52
Figura 31. Imagen en color 52

Figura 32. Modelo Pin-Hole 53
Figura 33. Nube de puntos en color desalineada 54
Figura 34. Nube de puntos en color alineada 55
Figura 35. Interfaz *rqt_gui* con los offsets activos 56
Figura 36. Esquema del comportamiento de la aplicación 56
Figura 37. Vista del conductor en el vehículo inteligente 57
Figura 38. Detección de la cara 58
Figura 39. Filtro *VoxelGrid* 58
Figura 40. Visualizador cara 61
Figura 41. Antes y después de aplicar matriz de rotación 62
Figura 42. Diagrama de flujo del funcionamiento de la aplicación 63
Figura 43. Fragmento de *rqt_graph* con la aplicación en marcha 64
Figura 44. Posición inicial 67
Figura 45. Movimiento Pitch 68
Figura 46. Movimiento Yaw 68
Figura 47. Movimiento Roll 69
Figura 48. Vista de Kinect y capturas lateral y frontal del conductor, respectivamente 70
Figura 49. Nube de puntos ligeramente rotada y ejes de coordenadas 71
Figura 50. Ordenador del vehículo y su ubicación 72
Figura 51. Visualizador movimiento Pitch 73
Figura 52. Gráficas movimiento Pitch 74
Figura 53. Visualizador movimiento Yaw 74
Figura 54. Gráficas movimiento Yaw 75
Figura 55. Visualizador movimiento Roll 76
Figura 56. Gráficas movimiento Roll 77
Figura 57. Funcionamiento con exceso de iluminación 80
Figura 58. Funcionamiento con visibilidad nula 81
Figura 59. Falso positivo en movimiento Yaw 82
Figura 60. Gráfica falso positivo en movimiento Yaw 83
Figura 61. Vehículo inteligente IVVI 2.0 84

Diseño de aplicación basada en Kinect para vehículo inteligente

ÍNDICE DE TABLAS

Índice de tablas

Tabla 1. Horas de iluminación 70
Tabla 2. Offset en ejes de coordenadas 71
Tabla 3. Rango de funcionamiento 82
Tabla 4. Coste de software 87
Tabla 5. Coste de hardware 88
Tabla 6. Coste de personal 88
Tabla 7. Coste total del proyecto 89

Diseño de aplicación basada en Kinect para vehículo inteligente

ÍNDICE DE ECUACIONES

Índice de ecuaciones

(1) Coordenada de la imagen en el eje X 53
(2) Coordenada de la imagen en el eje Y 53
(3) Distancia entre un punto del modelo y otro de la nube de puntos actual 59
(4) Mínimos cuadrados 59
(5) Traslación 59
(6) Rotación en torno al eje X 59
(7) Rotación en torno al eje Y 59
(8) Rotación en torno al eje Z 59
(9) Matriz de transformación 59
(10) Error de entrenamiento 60
(11) Orientación movimiento Pitch 60
(12) Orientación movimiento Yaw 60
(13) Orientación movimiento Roll 60
(14) Rotación de la nube de puntos respecto del eje Z 61

Diseño de aplicación basada en Kinect para vehículo inteligente

Capítulo 1: Introducción

1.1. Motivación del proyecto

En el presente documento, se observa el comportamiento del meta-sistema operativo ROS Hydro [1], el cual ofrece muchas posibilidades para crear aplicaciones de bajo y alto nivel en el campo de la robótica. Destaca su forma de comunicar los distintos nodos mediante el uso de mensajes que permiten compartir la información, así como la gran variedad de herramientas propias de las que consta, que ayudan a analizar, visualizar o modificar los procesos en tiempo real.

En este Trabajo Fin de Grado, se estudiarán las señales recibidas por un sensor Kinect [2] (facilitado por el Departamento de Ingeniería de Sistemas y Automática) y se desarrollará una aplicación que solucione un problema propuesto.

1.2. Objetivo del proyecto

El propósito de este proyecto es crear una aplicación para vehículo inteligente (IVVI 2.0 [3], Intelligent Vehicle based on Visual Information), a través de la cual podamos detectar la orientación de la cara al enfocar directamente sobre el conductor. El trabajo deberá cumplir las siguientes especificaciones:

- Se utilizará ROS (versión Hydro o posteriores) como plataforma donde se llevará a cabo todo el proyecto. Para ello, necesitaremos tenerlo instalarlo en el sistema operativo Ubuntu 12.04 LTS o versiones superiores [4].
- Los datos serán captados mediante un sensor Kinect, el cual se ubicará a una distancia de entre 0,60 m y 0,70 m del individuo.
- Mínimo tienen que existir 2 nodos o ejecutables los cuales deben comunicarse entre sí (en este caso, se han creado 3 nodos perfectamente enlazados, en los que cada uno realizará una función diferente en la aplicación).
- Los ejecutables serán implementados en lenguaje C++ o Python (se utilizará C++).
- Se deberá trabajar con imágenes y nubes de puntos. Obligatoriamente, su tratamiento se llevará a cabo mediante el uso de librerías OpenCV [5] y Point Cloud Library [6].
- Se tendrá que desarrollar un entorno de visualización para mostrar por pantalla las representaciones extraídas.
- Los datos se guardarán en ficheros de texto para posteriormente analizarlos.

En todo momento, se tendrá que asegurar el correcto funcionamiento de la aplicación, desarrollando un entorno de fácil uso y comprensión para todos los usuarios. Además, los ejecutables deberán estar perfectamente comentados y

estructurados, facilitando su manipulación para ser utilizados en posteriores aplicaciones de mayor nivel.

Esta aplicación ayudará al Departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid a llevar a cabo próximos proyectos en el vehículo inteligente.

1.3. Estructura y resumen del proyecto

En este apartado, se pueden apreciar las partes de las que se compone el Trabajo Fin de Grado, así como un breve resumen de todos sus capítulos. El proyecto se estructura de la siguiente forma:

- **Introducción:** Breve presentación donde se observan la motivación que lleva a elegir de este proyecto, los objetivos previos a cumplir y un resumen por capítulos indicando los temas principales abordados en el documento.
- **Estado del arte:** Se hará mención a todos aquellos conceptos necesarios para ponernos en la situación del proyecto. Comenzará explicando el significado de visión artificial y su evolución a lo largo de los años. Posteriormente, se nombrará su aportación a la automoción en los últimos años, así como los principales avances en este campo. Tras introducir estos temas, se comentarán los distintos modelos que se utilizan para analizar el entorno y los sistemas de detección facial. Finalmente, se hablará acerca las actuales técnicas y tecnologías de modelado 3D y adquisición de datos.
- **Software y hardware utilizados:** Herramientas informáticas y periféricos utilizados, aclarando sus características generales y especificaciones técnicas. En este punto, se abordan una serie de temas que son de gran importancia antes de comenzar la construcción del proyecto.
- **Diseño y desarrollo del proyecto:** En primera instancia, se verá una pequeña introducción sobre el problema propuesto a solucionar. Tras analizar esta información, se explicará en detalle en qué consisten los 3 nodos o ejecutables de los que se compone la plataforma (adquisición de la nube de puntos, detección de la cara y su orientación, y visualización de los resultados). Por último, se hará referencia a la puesta en marcha de todo el sistema.
- **Resultados prácticos:** Se estudiarán de forma detenida los datos obtenidos en archivos de texto y se procederá a analizar los resultados apoyándonos de gráficas que facilitarán su comprensión. Para ello, habrá que realizar varias pruebas las cuales se basarán en 3 movimientos diferentes de la cara y se observará cómo se comporta la aplicación y de qué forma varía la orientación en cada caso. También, cabe destacar que se marcarán unos

parámetros iniciales utilizados a la hora de extraer los resultados (iluminación y posición/orientación del sensor).

- **Conclusiones:** Se destacarán los aspectos más relevantes del trabajo, las dificultades encontradas y posibles mejoras que se podrían realizar, y se explicarán las aplicaciones futuras para las que ha sido creado.
- **Presupuesto:** Costes totales de personal, equipos y materiales necesarios.
- **Bibliografía:** Referencias a la documentación utilizada para el desarrollo del proyecto.

Diseño de aplicación basada en Kinect para vehículo inteligente

Capítulo 2: Estado del arte

2.1. La visión artificial y su evolución histórica

Primeramente, deberemos mencionar a qué nos referimos cuando hablamos del término *sentido*, según la RAE, se trata de "Proceso fisiológico de recepción y reconocimiento de sensaciones y estímulos que se produce a través de la vista, el oído, el olfato, el gusto o el tacto, o la situación de su propio cuerpo". De la misma forma, nos interesa conocer el significado de *vista*, "Sentido corporal con que se perciben los objetos mediante la acción de la luz". Igualmente, la *visión* se define como "Acción y efecto de ver". [7]

Una imagen es una representación bidimensional del entorno percibido por una cámara, compuesta por píxeles (unidad más pequeña de color, el cual tiene valores comprendidos entre 0 y 255). Un video es una secuencia de imágenes que se visualizan con una determinada velocidad que viene dada en fps (capturas por segundo).

En los seres humanos, la vista es el sentido más complejo y con mayor importancia debido a que la realidad captada por sus receptores (2 millones de fibras nerviosas), suponen las tres cuartas partes de la información procesada por el cerebro [8]. Los órganos encargados de la visión son los ojos (Véase Figura 1), a través de los cuales podemos reconocer personas u objetos, así como su posición y orientación en el entorno, detectar movimiento, obstáculos o presencia de posibles peligros, analizar colores y formas, etc. El cerebro estudia las señales luminosas percibidas por el ojo y se encarga de ordenar e interpretar la información recibida, permitiendo extraer las características de la imagen. [9]

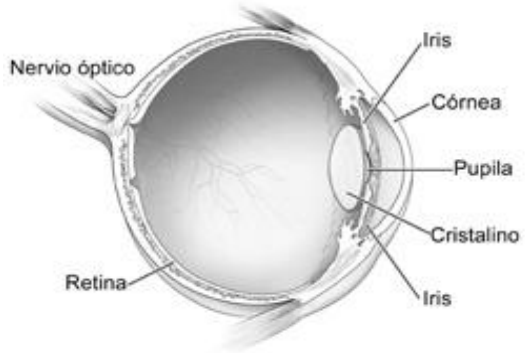


Figura 1. El ojo humano.

El término de *visión artificial* (también llamado en la actualidad *visión por computador*) se refiere al análisis de las imágenes que son capturadas por una cámara, mediante el uso de computadores que nos permitan extraer la descripción de los objetos físicos percibidos. Existe cierto parecido entre la visión humana y la visión artificial, ya que el papel que desempeña el ojo lo llevará a cabo una cámara, mientras que el computador realizará las labores del cerebro, aunque hay que tener en cuenta que no se pretende imitar el sentido de la vista. [10]

Durante los años cincuenta, tras la aparición de los primeros computadores, en los que se tenía gran confianza, se pensaba que no sería complicado desarrollar el sentido de la vista de forma artificial. Esta mentalidad surgió debido a que "ver" es una tarea que las personas realizan fácilmente y sin dificultades, por lo tanto para un ordenador también debería ser sencillo. Fue entonces cuando aparecieron los primeros trabajos en este campo, en los que se utilizaron cámaras para la percibir las características del entorno. En 1963, Roberts demostró que se podían obtener descripciones matemáticas al procesar imágenes digitalizadas, mediante transformaciones homogéneas, de donde extraía la posición y orientación de los objetos físicos capturados. En 1966, Marvin Minsky propuso como proyecto que un computador describiera lo que viese, pero nunca se llegó a finalizar. Fue posteriormente, en 1967, cuando Wichman desarrolló un sistema compuesto por una cámara de televisión conectada a un ordenador, que permitía identificar la posición y forma de los objetos en tiempo real. Entre 1965 y 1970, Roberts, Sobel y Prewitt llevaron a cabo 3 modelos diferentes de detección de bordes en las imágenes, los cuales siguen siendo utilizados en las tecnologías actuales (Véase Figura 2). [11]

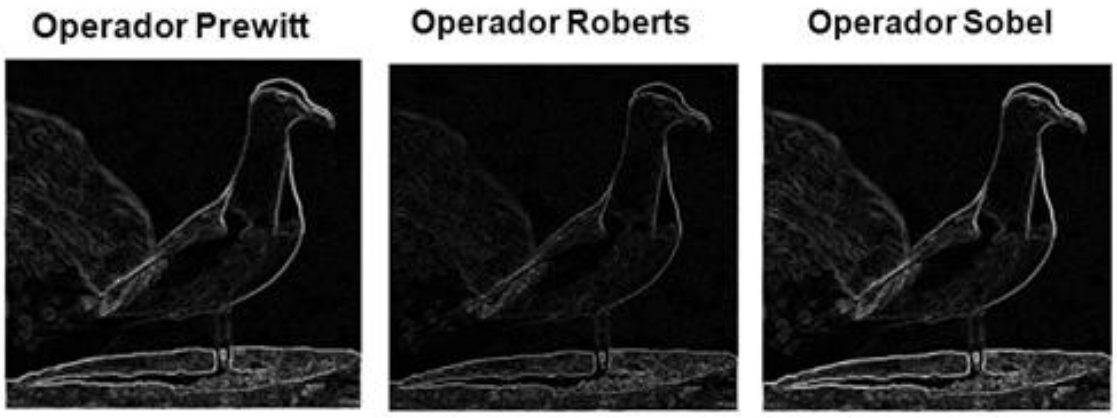


Figura 2. Métodos de detección de bordes.

Tras esta etapa, donde no se obtuvieron los avances esperados, hubo un periodo de reflexión donde se paralizaron las investigaciones en esta rama. Fue a partir de la década de los ochenta, donde volvieron a aparecer nuevos estudios que se centraron en la extracción de características. Destacan los primeros trabajos sobre detección de texturas (Haralik, 1982), esquinas (Kitchen, y Rosendfeld, 1982, Véase Figura 3), líneas (Kanade) y formas (Steven y Witkin, 1981), así como las primeras manifestaciones de detección de movimiento (Horn) y visión estéreo (Mayhew y Frisby). A partir de esta época, ha surgido un gran interés en la visión artificial, gracias a la aparición de los primeros congresos internacionales y revistas especializadas o a la incorporación de los avances en los planes de estudios de muchos centros educativos, entre otros. Hoy en día, los principales objetivos en este campo se centran en el procesamiento de imágenes (transformación de una imagen en otra), la generación de gráficos por ordenador (por ejemplo, análisis de muestras mediante histogramas) y reconocimiento de objetos (en base a unos patrones definidos). [11]

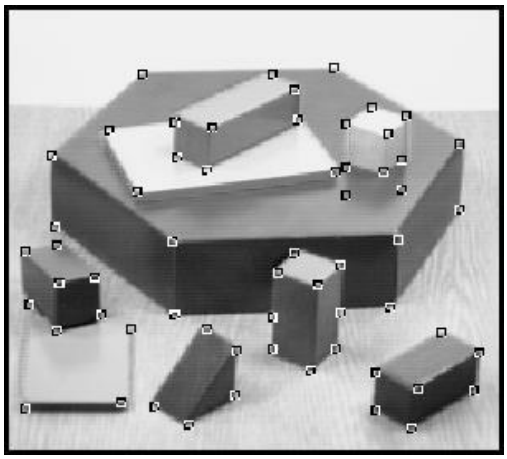


Figura 3. Método de detección de esquinas.

2.2. Visión artificial en la automoción

Tras conocer los principales conceptos de la visión artificial, se procederá a explicar las principales aplicaciones de esta rama en la automoción. Durante los últimos años, han sido muy escasos los sistemas de detección utilizados para este fin, aunque las investigaciones en este campo avanzan a pasos agigantados. Estas herramientas se utilizan sobre todo para dos objetivos:

- Realizar un control del tráfico mediante cámaras implementadas sobre postes altos o farolas (Véase Figura 4). Su finalidad es visualizar en tiempo real lo que sucede en las carreteras para detectar atascos, accidentes o negligencias por parte de los automovilistas.

Figura 4. Cámara de tráfico.

- Reconocimiento e identificación de las matrículas de los vehículos en caso de excesos de velocidad. Estos equipos suelen acompañar a radares (Véase Figura 5), tanto fijos como móviles, para capturar una imagen de aquel que ha cometido la infracción.

Figura 5. Radar fijo con cámara.

Una de las casas que más empeño está poniendo en aplicar estas tecnologías en sus vehículos es la marca Nissan. Se ha centrado en las necesidades de los conductores intentando reducir el punto ciego del conductor (Véase Figura 6). Por este motivo, ha creado una cámara trasera para motorizar lo que ocurre a los laterales y detrás del vehículo, mostrándolo por el ordenador de a bordo. Está compuesto por múltiples sensores con cámara, así como advertencias visuales y sonoras para avisar en todo momento al sujeto. En la actualidad, existen automóviles de otras marcas (por ejemplo, Peugeot) que incorporan sensores traseros para ayudar a aparcar, pero Nissan ha llegado más lejos al poder detectar y visualizar posibles situaciones de peligro en la zona trasera y en los laterales del coche. [12]

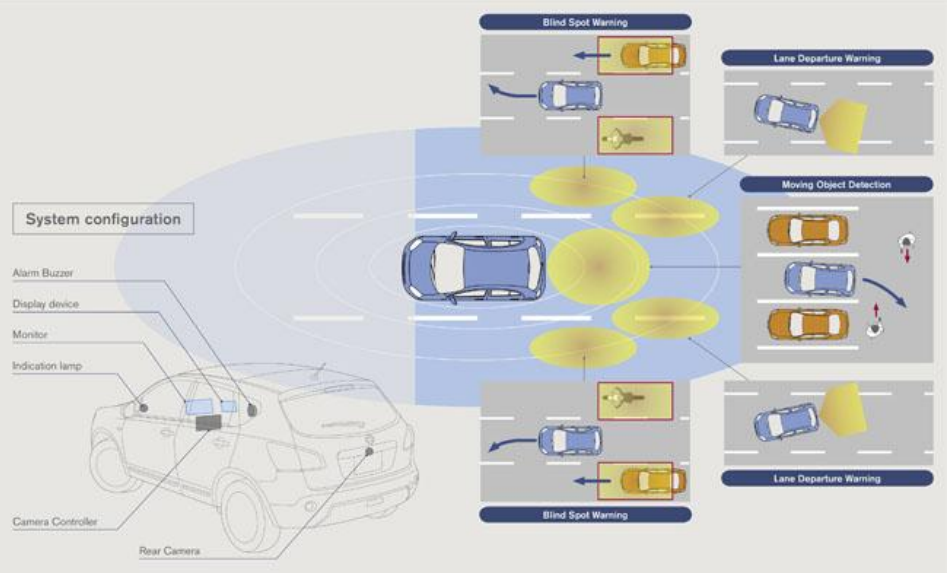


Figura 6. Sistema de detección de punto ciego Nissan.

Otras aplicaciones para las que se usa la visión artificial son: inspección de las llantas de los coches y de la soldadura de las piezas, basadas en reconocimiento 3D; funcionamiento de Luneta Térmica Antivaho, utilizando cámaras térmicas; revisión de la tapicería y medición de piezas mecánicas y plásticas, mediante cámaras de alta precisión; y análisis de los test de alta velocidad, que necesitan cámaras capaces de captar datos de forma muy rápida (500 imágenes/segundo) [13]. Realmente, estos sistemas están siendo usados en los controles de calidad para asegurar que no haya ningún tipo de problema en el vehículo.

Actualmente, las principales líneas de investigación se están centrando en estudiar lo que ocurre tanto dentro como fuera del vehículo, por ejemplo monitorizar al conductor, detección de peatones y de señales o reconocimiento de los carriles de las carreteras, entre otros.

2.3. Modelos de análisis del entorno

Para estudiar el comportamiento de determinados elementos o características de una imagen o video (secuencia de imágenes), será necesario realizar un análisis completo del entorno extraído. Principalmente, las pautas que se siguen en este campo son:

1. Procesamiento y extracción de características: Consiste en resaltar alguna característica significativa de la imagen. Destaca el realce de contornos mediante la utilización de filtros paso alto o hacer pasar a la imagen por

filtros paso bajo para eliminar el posible ruido que exista en ella. También en este punto se pueden incluir la detección de bordes, movimiento o incluso texturas. [11]

2. Separación del fondo: Una vez resaltadas las características principales, lo que se pretende es eliminar aquello que se considera fondo, es decir, todo lo que no sea nuestro objeto de estudio. Para ello, cabe destacar los métodos como la Transformada de Hough [14], la cual consiste en utilizar formas geométricas para encontrar unos parámetros deseados en la imagen a partir de una detección de bordes, o la segmentación por el sistema Split & Merge [15], que separa en regiones la muestra, para después unir aquellas partes que tienen relación entre ellas. Con esto, solo se extraería la información que se quiere analizar, por lo tanto, estos procesos aumentan la velocidad de procesamiento.

   En la mayoría de las ocasiones, se tienen que aplicar transformaciones morfológicas para ajustar de alguna forma lo detectado a las necesidades del estudio, como por ejemplo aplicar una erosión [16] o dilatación [17] (disminuir o ampliar la señal extraída tras la separación del fondo, respectivamente) o aplicar los Descriptores de Fourier [18], permitiendo obtener la forma de un objeto a partir de una muestra binaria.
3. Reconocimiento de patrones: Se basa en la utilización de diversos procedimientos que permitan reconocer un objeto en base a una serie de patrones marcados. Actualmente, es importante recalcar algoritmos utilizados para la identificación de movimientos, como por ejemplo el empleado en el proyecto. Este método se basa en la detección de los píxeles vecinos más próximos entre dos capturas [19] para hacer un seguimiento y extracción del rostro.

En el caso de tratarse de muestras en 3D, uno de las principales transformaciones que se hacen es utilizar un filtro que divide la imagen en pequeños cubos tridimensionales. De ella se extrae una nueva señal formada por todos los centroides (centros de masas) de cada uno de estos hexaedros para todos los puntos. Este filtro se conoce como *voxel* [20] y nos permite analizar un número inferior de puntos con un menor coste computacional.

2.4. Sistemas de detección facial

La detección de rostros es una tarea de difícil ejecución para las computadoras, debido a que existen muchos tipos de caras, así como dificultades como escala, orientación, posición, expresiones faciales, iluminación, etc. En la actualidad, el reconocimiento del rostro es una de las principales líneas de investigación en el campo de la visión artificial. Se pueden encontrar en sistemas de

seguridad (por ejemplo, frente al terrorismo), en la automatización de las viviendas (por ejemplo, en apertura de puertas) o en servicios de entretenimiento, entre otros. Primeramente, estos programas necesitan de una captura en video o imagen para poder extraer sus características del fondo y proceder a su posterior análisis. En estos últimos años, cabe destacar que han aparecido diferentes métodos de identificación facial basadas en el uso de diferentes técnicas. [21]

En 1999, Hjelmas y Wroldsen, realizaron un reconocimiento de rostro basado únicamente en un estudio de los ojos. Para ello, tomaron como referencia el centro de la distancia existente entre los dos ojos, y estimaron el tamaño de la cabeza para terminar dibujando un rectángulo que comprendiera el rostro. Este proyecto generó buenos resultados, pero al hacer una evaluación aproximada del comportamiento de la cara, generó muchos datos falsos. [22]

Otro caso a destacar, fue el creciente interés por estudiar la nariz. Por este motivo, en 2006 apareció el primer algoritmo que reconocía en 3D el rostro a partir del análisis de las diferentes regiones de la nariz frente a diferentes gestos. Aunque necesita algunas mejoras, la respuesta del sistema ha sido muy satisfactoria, siendo comprobada en más de 400 personas. [23]

Sin embargo, en el año 2001, Viola y Jones [24] crearon un algoritmo que procesaba imágenes de forma extremadamente rápida con una alta precisión de detecciones faciales en tiempo real. Este sistema se basa en 3 partes [25]:

- Extracción de características de tipo *Haar* [24]: En un marco de detección y considerando regiones rectangulares contiguas (Véase Figura 7), se suman los valores de los píxeles en cada región y posteriormente se restan los de la zona oscura con los de la zona blanca. De esta forma, se separan las diferentes secciones de la imagen, la cual se denomina *Imagen Integral*.

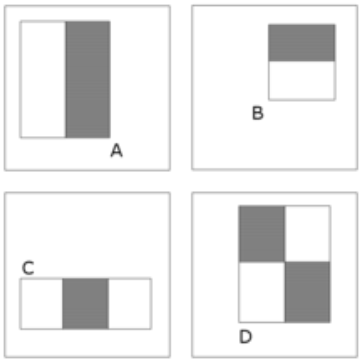


Figura 7. Regiones rectangulares.

- Etapa de aprendizaje: Se marcan unos patrones positivos (coinciden con el objeto) y negativos (que no se relacionan con el objeto) a una serie de filtros en cascada para que se forme un modelo de características del tipo *Haar* (denominadas *débiles*, ya que por separado no son suficientes para garantizar el reconocimiento).
- Cascada de características: Una vez tengamos la cascada de características, se podrán determinar cuáles de ellas se corresponden con el objeto a detectar, y aquellas que cumplen los mismos requisitos se agrupan para indicar que se trata del objeto.

Para asegurar una mejor compresión, en la Figura 8, se puede ver un esquema del funcionamiento de este algoritmo, donde cada F es una característica del tipo *Haar*:

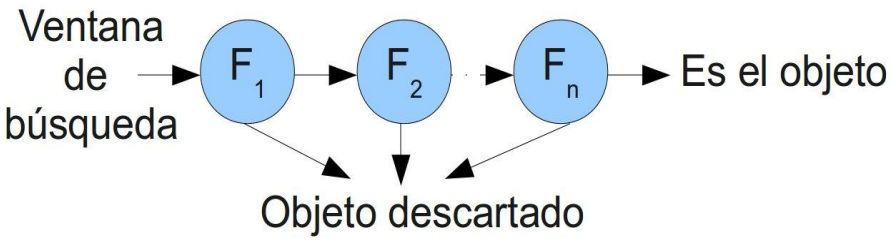


Figura 8. Esquema de funcionamiento Algoritmo Viola-Jones.

Esta aplicación utiliza este procedimiento para realizar el reconocimiento de la cara, gracias a que la biblioteca OpenCV nos permite tener al alcance estas herramientas. Actualmente, es el método más usado y el que asegura una precisión mayor, teniendo una probabilidad muy próxima al 0% de que ocurra un falso negativo y un porcentaje cercano al 40% de que se produzca un falso positivo. [25]

2.5. Modelado y tecnologías de adquisición de datos 3D

Existen distintos tipos de modelado, o construcción de objetos en 3D, en función de las necesidades y del tipo de información recibida por el sensor. Los principales tipos son:

- Nubes de puntos: Consiste en la captación de un conjunto de puntos, los cuales son representados en el entorno en función de sus coordenadas tridimensionales en el espacio. A una nube de puntos que abarca un entorno bastante amplio se le puede denominar mapa. Por lo tanto, el mapa final dependerá de la posición, orientación y profundidad de los objetos en la realidad.

También, cada punto puede tener incorporado sus propios valores de color (Véase Figura 9). Para esto es necesario realizar una correlación entre datos de color (extraídos de la imagen) y profundidad.

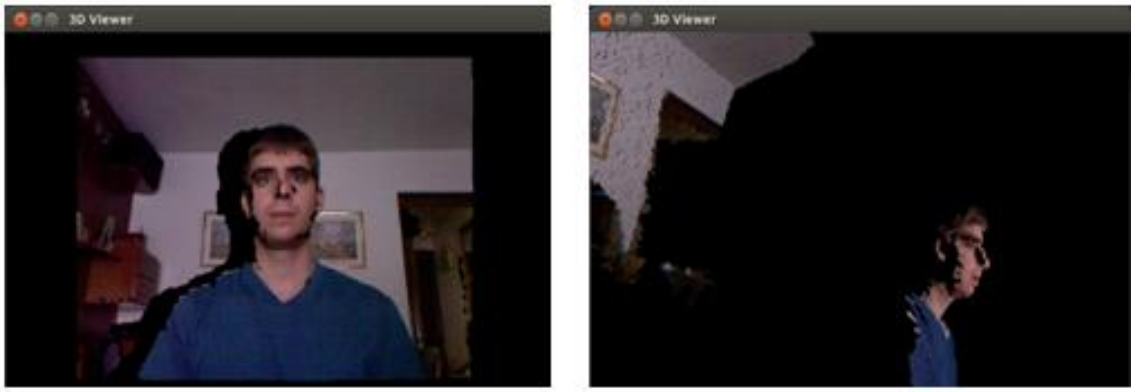


Figura 9. Vista frontal y lateral de una nube de puntos en color.

- Poligonales: Consiste en la representación de un objeto en 3D a partir de superficies planas. Por este motivo, el modelado de superficies curvas se llevará a cabo mediante la unión de una gran cantidad de superficies poligonales. Suelen tener resoluciones muy bajas. [26]

- Spline: Un *spline* es una línea (generalmente curva), que va siguiendo unos determinados puntos de control. Este modelado consiste en obtener una representación en 3D a partir de la unión de pequeñas superficies curvas. Existen diferentes tipos de superficies, como por ejemplo el B-spline, Bezier o NURBS, entre otros. Cabe destacar que este tipo de modelado es independiente de la resolución, por lo tanto no se pierde información al modificar la escala o el zoom. [27]

- Paramétricos: Es un método utilizado para la creación de sólidos a partir de sus parámetros principales (altura, anchura y profundidad), los cuales se almacenan y permanecen hasta que se produce una modificación que deforma el objeto. No obstante, al capturar los datos se puede volver a un estado anterior a la deformación. Normalmente, se basa en la utilización de la técnica *spline*, pero no todos los modeladores *spline* son paramétricos. [27]

- CAD: Actualmente, estos sistemas son los más avanzados y más utilizados en la ingeniería debido a que incorporan tanto la representación 3D como la información referente al objeto que estamos modelando. Incluso, algunos sistemas CAD tienen internamente simuladores dinámicos para estudiar comportamientos de los materiales (por ejemplo, pruebas de flexibilidad o tracción de una pieza). [28]

Actualmente, las principales investigaciones en el campo de la visión artificial utilizan como herramientas 3D sensores que permiten percibir y reconstruir objetos a partir de la realidad. Por este motivo, se presentan una serie de sensores que trabajan mediante diferentes sistemas de captación de datos: cámara de tiempo de vuelo, cámara estéreo, Kinect.

- Cámara de tiempo de vuelo: Estos escáneres crean una representación tridimensional al medir las distancias a partir del tiempo que tarda un pulso de luz en chocar contra el objeto y retornar para ser detectado por el receptor (Véase Figura 10). Se trata de sensores de alta precisión con un rápido muestreo. [29]

Figura 10. Cámara de tiempo de vuelo.

- Cámara estéreo: Esta tecnología utiliza dos lentes que, mediante la extracción y superposición de dos imágenes bidimensionales captadas desde puntos de vista diferentes, permiten extraer una tercera dimensión, la profundidad. Lo que se pretende es simular el funcionamiento de la vista humana, la cual percibe la luz de los objetos con cada ojo y posteriormente, el cerebro interpreta estas dos señales para crear una representación tridimensional del entorno. [30]

- Sensor Kinect: Este equipo (Véase Figura 11) incorpora dos cámaras: una que percibe el color de los objetos y otra que interpreta la profundidad. Para este último caso, un emisor lanzará rayos infrarrojos y se creará una nube de puntos en función de la señal captada por el receptor. Este sensor, permite correlacionar color y profundidad de forma simultánea para crear un mapa tridimensional con datos RGB.

Figura 11. Sensor Kinect.

También, existen otro tipo de escáneres que necesitan mantener un contacto con el objeto para construir un modelo tridimensional. Estos sensores son los llamados CMM (Máquina de medición por coordenadas) [31] y permiten obtener las medidas de los objetos con solo palpar su superficie con una punta de acero duro o zafiro.

Para el proyecto, se ha elegido un sensor Kinect ya que se trataba de un sistema de bajo coste que se ajustaba a las necesidades de la aplicación al poder percibir datos de profundidad y color en tiempo real. Gracias a este equipo se podrán modelar nubes de puntos y se aplicarán los algoritmos necesarios para poder trabajar con ellas.

Diseño de aplicación basada en Kinect para vehículo inteligente

Capítulo 3: Software y hardware

3.1. ROS

ROS (Véase Figura 12), Robot Operating System, es un framework creado por Willow Garage para plataformas robóticas. Es un meta-sistema operativo con licencia BSD (código abierto), que proporciona los servicios de un sistema operativo, incluyendo control de dispositivos, funcionalidad y reusabilidad (puede permitir trabajar incluso en tiempo real), comunicación entre los procesos (también conocidos como nodos) por medio de mensajes y una gestión por medio de paquetes. Se trata de un software libre que incorpora multitud de bibliotecas y herramientas que facilitan el desarrollo aplicaciones [32]. Además, en su página web [33] se pueden encontrar paquetes ya creados por otros usuarios para algunos robots específicos, así como diversos tutoriales [34], documentación API y ejemplos para comprender su funcionamiento.

Figura 12. Logo ROS (Robot Operating System).

Este framework está siendo construido y modificado por el esfuerzo conjunto de la comunidad robótica, por lo tanto los avances de la plataforma está en manos de los usuarios. Además cuenta con un foro, denominado *ROS Answers* [35], donde podemos obtener respuesta a muchas cuestiones acerca de su comportamiento para diferentes proyectos y situaciones.

Las diferentes versiones que existen de ROS y sus fechas de divulgación [32] son:

- ROS Indigo Igloo [36]: Julio de 2014.
- ROS Hydro Medusa [1]: Septiembre de 2013.
- ROS Groovy [37]: Diciembre de 2012.
- ROS Fuerte [38]: Abril de 2012.
- ROS Electric [39]: Agosto de 2011.
- ROS Diamondback [40]: Marzo de 2011.
- ROS C Turtle [41]: Agosto de 2010.
- ROS Box Turtle [42]: Marzo de 2010.

3.1.1. Información general. ROS Hydro Medusa

ROS Hydro Medusa (Véase Figura 13) fue lanzado el 4 de Septiembre de 2013. Es la séptima versión de ROS, aunque no la última y en ella se han centrado en la conversión de muchos paquetes anteriores al nuevo sistema *catkin* [43]. Está dirigida principalmente para Ubuntu 12.04 LTS o versiones superiores, pero también puede ser compatible con otros sistemas de Linux, MAC, Android o Windows, entre otras, pero con diversas limitaciones. [44]

Figura 13. Logo ROS Hydro Medusa.

Su anterior versión (ROS Groovy [37]) fue la primera en incorporar *catkin*, puesto que las plataformas previas usaban otro sistema diferente denominado *rosbuild* [45]. Lo que se ha querido conseguir es un sistema más sofisticado con un número mayor de aplicaciones y con una mejora del software en términos de herramientas y componentes, e incluso está promoviendo y facilitando la migración de *rosbuild* a *catkin*.

3.1.2. Objetivos de ROS

Actualmente, la comunidad robótica pretende unificar y estandarizar todo tipo de aplicaciones de bajo nivel con el fin de facilitar la obtención de aplicaciones de mayor nivel. Esto tiene como fin promover el desarrollo de algoritmos que permitan trabajar con una mayor autonomía a los equipos y robots.

Este software permite la reutilización de una colección de algoritmos comunes para diferentes tareas. Por este motivo, muchas universidades, instituciones o usuarios ponen a disposición de cualquier persona de forma libre una serie de paquetes que puedan ayudar a crear nuevos programas, siempre con unos objetivos finales definidos. De esta forma, se puede reducir el tiempo de construcción de nuevas plataformas a partir de otras ya creadas, minimizando el número de errores

y evitando tener que repetir pruebas realizadas y comprobadas por otros usuarios. Por lo tanto, el único objetivo de ROS es poner en contacto los trabajos de varios usuarios para únicamente avanzar en esta rama. Cabe destacar que, en el campo de la visión artificial, ROS ha estandarizado el tipo de mensajes pudiendo utilizar una misma configuración para diferentes robots. También, proporciona una forma sencilla de reproducir y grabar las señales recibidas de varios sensores. [46]

Este framework se encuentra en un rápido proceso de crecimiento en la investigación robótica. Además, está siendo construido y mejorado gracias a la contribución de científicos, investigadores y demás usuarios que colaboran en el avance de este campo.

3.1.3. Conceptos generales

ROS está compuesto por una serie de conceptos en función del nivel [47] del que estemos tratando:

- Nivel de sistema de archivos: Se refieren a los recursos que utiliza ROS.
  - Paquetes [48]: Son la unidad principal de la organización de ROS. Cada uno de ellos puede contener diferentes procesos (nodos o ejecutables), bibliotecas, bases de datos o archivos de configuración. Son el elemento de construcción y liberación de este software.
  - Meta-paquetes [49]: Se trata de paquetes especializados que únicamente se utilizan para representar un grupo de paquetes interrelacionados.
  - *package.xml* [50]: Proporciona metadatos acerca de un paquete (nombre, versión, información de la licencia o dependencias, entre otros).
  - Repositorios [51]: Es un conjunto de paquetes que comparten un sistema VCS (sistema de comunicación digital) común. También, pueden estar formados por un solo paquete.
  - Tipos de mensajes (*msg*) [52]: Estructuras de datos de los mensajes que son enviados en ROS.
  - Tipos de servicios (*srv*) [53]: Estructuras de datos de solicitud y respuesta de los servicios ROS.
- Nivel de computación gráfica: Constituyen la red de procesos que se están ejecutando en ROS.
  - Nodo [54]: Ejecutable o proceso de este software que se utiliza para el intercambio de información. Un sistema suele contener más de un nodo que se comunican entre sí.

- Maestro [55]: Proporciona las herramientas de ROS, sin él ningún nodo podría comunicarse mediante mensajes o establecer servicios.
- Servidor de parámetros [56]: Actualmente, forma parte del Maestro, pero su acometido es permitir que los datos sean almacenados.
- Mensajes [57]: Estructura de datos que utilizan los nodos o ejecutables para establecer una comunicación en el sistema.
- Topics o temas [58]: Sistemas de transporte que usa ROS para que los nodos puedan mandar y recibir los mensajes. De esta forma, cuando se quiera captar un mensaje habrá que subscribirse a un topic, mientras que cuando queramos enviarlo habrá que volcar la información sobre un tema para publicarlo. Solamente se podrá acceder al mensaje si nos subscribimos en el formato correcto.
- Servicios [59]: Se definen por un par de estructuras de mensajes, uno para solicitud y otro para la respuesta, es decir, un nodo ofrece un servicio mediante un mensaje de solicitud y espera una respuesta.
- Bolsas (*bags*) [60]: Formato para almacenar y reproducir las estructuras de datos de ROS.

Para comprender mejor estos conceptos, en la Figura 14 se podrá observar un esquema simple del proceso de computación de ROS:

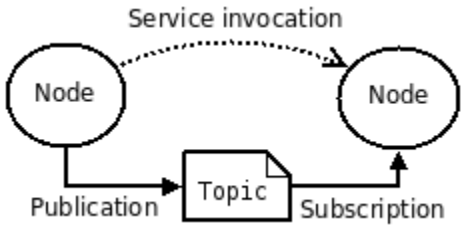


Figura 14. Funcionamiento de ROS.

- Nivel de la comunidad de ROS: Son aquellos recursos que permiten a la comunidad robótica el intercambio de programas y conocimientos. Estos recursos son:
  - Distribuciones [61]: Conjuntos de *stacks* o pilas [62].
  - Repositorios [51]: Red de repositorios de código que puedan ser reutilizables o adaptables para diferentes robots.
  - *ROS Wiki* [63]: Donde se encuentra la información de ROS. Cualquier persona puede contribuir a modificar, actualizar o completar información.
  - *Bug Ticket System* [64]: Sistema para sugerencias y avisos de errores del software.

- Listas de correo [65]: Es el canal principal de comunicación sobre los principales cambios de este framework.
- *ROS Answers* [35]: Foro donde los usuarios pueden preguntar acerca de dudas relacionadas con este meta-sistema operativo.
- Blog [66]: Se trata de un blog compuesto por documentación sobre ROS, así como imágenes y videos que muestran ejemplos de uso.

3.1.4. *catkin* vs *rosbuild*

Tras la dificultad que tenía *rosbuild* para poder ser utilizado en otros sistemas operativos dirigidos por Willow Garage, obligó a modificar esta plataforma para crear un nuevo sistema basado en una arquitectura de paquetes (*catkin*) que solucionara este problema. No obstante, la compañía ha facilitado la conversión del sistema anterior al actual para que proyectos realizados a partir de *rosbuild* no se perdieran o quedaran inutilizables. Las principales diferencias [67] entre ellos son:

- Reducción de la velocidad de construcción con *catkin*, ya que permite trabajar con un número menor de dependencias.
- *rosbuild* se compone de *stacks* (pilas) y paquetes. Un paquete es una carpeta que contiene un *manifest.xml* [50], un Makefile acompañado de un *CMakeLists.txt* [68]y archivos, mientras que una pila es una carpeta con un archivo *stack.xml*. Las pilas estaban formadas por paquetes y solo dependían de otras pilas, de la misma forma que los paquetes solo dependían de otros paquetes. Por otro lado, *catkin* se compone de paquetes, los cuales se componen de los diferentes archivos, un *CMakeLists.txt* [69] (se elimina el Makefile) y un *package.xml* [50] (en lugar del *manifest.xml* de *rosbuild*, teniendo una sintaxis diferente), y meta-paquetes (que sustituyen la función de las pilas).
- En cuanto al proceso de construcción, *catkin* utiliza un espacio de trabajo que se tendrá que crear, y se compone de los comandos *Cmake* estándar donde se encontrarán las dependencias utilizadas. Para cada nuevo paquete se creará un nuevo archivo *Cmake* con el fin de construir una cadena de herramientas. Por este motivo, se puede acceder a todos los meta-paquetes y poder compilar archivos en paralelo de proyectos interrelacionados. Sin embargo, en *rosbuild* esta compilación habría que realizarla secuencialmente. Únicamente, en este proceso pueden surgir varios conflictos como pueden ser la repetición de nombres de variables globales o de los ejecutables que puede dar lugar a errores de compilación, por lo tanto habría que evitar estas situaciones.
- En cuanto al empaquetamiento, los paquetes en *rosbuild* se instalan en una sola carpeta, es decir, que todos los paquetes de una pila se empaquetan en

una única unidad de distribución. Por el contrario, los paquetes de *catkin* se encuentran repartidos en diferentes carpetas, permitiendo tener varios ejecutables locales. De esta forma, esta distribución en subcarpetas puedan facilitar la reusabilidad de estos paquetes.

- *rosbuild* permite tener los paquetes en desarrollo en más de un espacio de trabajo, mientras que *catkin* se limita a un solo espacio de trabajo que debe contener un archivo *Cmake*. Además, permite separar en una carpeta el proceso de construcción para facilitar la compilación cruzada, aumentando de esta forma la velocidad de ejecución con respecto a *rosbuild*. De igual forma, el flujo de trabajo se basa en otra única carpeta compuesta de un archivo *CMakeLists.txt* y los nodos. También incluye una carpeta *devel* que se refiere a las carpetas con archivos de fuente (*.sh*) para la configuración del espacio.

Como se puede observar, el paso de *rosbuild* a *catkin* ha supuesto una mejora en ROS, tanto a modo en la construcción como en la ejecución, permitiendo reducir el coste computacional y adaptando esta herramienta para una gran variedad de sistemas operativos.

3.1.5. Librerías

En este apartado, hablaremos sobre las librerías que usaremos en el desarrollo de la aplicación. Aparte de aquellas que incluyen las sentencias de ROS Hydro, necesitaremos otras que nos permitirán el tratamiento de las imágenes y nubes de puntos captadas por el sensor Kinect: Point Cloud Library [6] y OpenCV [5].

3.1.5.1. Point Cloud Library

Point Cloud Library (PCL, Véase Figura 15) es una biblioteca que incorpora ROS para el tratamiento de nubes de puntos. Se trata de una librería gratuita con licencia BSD plenamente desarrollada para lenguaje C++, que incorpora diferentes algoritmos que se utilizan para el desarrollo de diferentes técnicas de manipulación en 3D, como por ejemplo filtrado, extracción de características o segmentación, entre otros. Uno de los principales sensores que se utilizan para este procesamiento es Kinect, que se creó para la consola XBOX 360, pero que ofrece muchas aplicaciones para trabajar con nubes de puntos en tiempo real, así como capturas bidimensionales. También, cabe destacar que PCL se divide en una serie bibliotecas que incluyen sentencias y códigos más pequeños para simplificar el desarrollo y permitir compilar por separado. Además, puede capturar nubes de puntos guardándolas en archivos *.pcd* para permitir un posterior análisis de la señal. [70]

Figura 15. Logo Point Cloud Library.

Esta biblioteca, es totalmente compatible con diferentes sistemas operativos, tales como Windows, MacOS, Linux o Android. Además, su página web [6], cuenta con una gran cantidad de tutoriales que son muy útiles a la hora de comprender su funcionamiento.

En cuanto a su funcionalidad con ROS, PCL ha creado unos plugins que funcionan como nodos de ROS llamados *nodelets* [71], que nos permiten acceder a los datos que extrae el sensor conectado (en el caso de este proyecto será Kinect). Por otro lado, esta librería incorpora su propia plataforma de visualización para mostrar por pantalla las nubes de puntos. [70]

3.1.5.2. OpenCV

OpenCV (Open Source Computer Vision Library) [5] es una biblioteca libre con licencia BSD, desarrollada por Intel en 1999, para el tratamiento de imágenes, videos y representaciones tridimensionales siendo utilizada en el campo de la visión por computador (Véase Figura 16). Se encuentra incluida en ROS por Willow Garage [72], cuenta con interfaces en diferentes lenguajes de programación (C, C++, Python y Java) y es compatible con Windows, Linux, MacOS, iOS y Android [73]. Además, en su página web [5] se puede encontrar toda la documentación API de las sentencias que incorpora.

Figura 16. Logo OpenCV.

Para poder trabajar con imágenes en ROS es necesario utilizar el módulo *CvBridge* [74] que transforma los mensajes de este software al formato de OpenCV (del tipo *CvMat* [75]). Esta adaptación facilita el manejo de muestras bidimensionales y tridimensionales en este meta-sistema operativo.

En este proyecto, esta biblioteca se utiliza para la manipulación del color de las imágenes y para realizar el *Modelo Pin-Hole* [11], atribuyendo los píxeles de la imagen en color a las nubes de puntos. Por otro lado, se usará para aplicar diferentes sentencias que colaboran la detección de rostros.

3.1.6. Lenguajes de programación. Lenguaje C++

El lenguaje de programación C++ es un lenguaje libre, creado en 1979 por Bjarne Stroustrup, como una ampliación del lenguaje C. Se caracteriza por la utilización de *clases* (agrupaciones de métodos y variables dentro de una misma identidad), concepto que para el anterior lenguaje C no existía [76]. Este modo de programación se caracteriza por el manejo de excepciones (tratamiento de errores), plantillas, sobrecarga de funciones o herencia, entre otros. Se trata de un lenguaje de programación sencillo, flexible, de fácil aprendizaje y con multitud de aplicaciones, por ejemplo tratamientos de imágenes, programación de bases de datos o realización de interfaces gráficas. Incorpora una serie de bibliotecas estándar y es fácilmente adaptable para poder utilizar otras librerías, como pueden ser OpenCV o Point Cloud Library. Este lenguaje es compatible en varios sistemas operativos, pero para cada uno de ellos se utilizaran compiladores diferentes, entre los que destacan:

- Windows: Dev C++, Visual C++.
- MacOS: Xcode.
- Ubuntu: Qt Creator.
- Linux: Eclipse.

Además, hay que hacer especial énfasis en que se trata de una programación orientada a objetos, es decir, que se utiliza un lenguaje en el que cada componente es considerado un objeto, con sus propios atributos y funciones. También, cabe mencionar que se puede mezclar perfectamente con el lenguaje C. [76]

3.1.7. Entorno de programación. QtCreator 2.3

Para implementar los nodos del sistema en lenguaje C++, se ha utilizado QtCreator [77] (Véase Figura 17) en su versión 2.3 para Ubuntu 12.04 LTS. Se trata de una plataforma IDE (entorno de desarrollo integrado), es decir, un editor de

texto avanzado que da soporte para la escritura de ejecutables en diversos lenguajes de programación (C++, QML y ECMA).

Figura 17. Logo QtCreator.

Este entorno de programación facilita la escritura del código ofreciendo soporte a la hora de escribir las sentencias, crear interfaces gráficas, diferenciar por colores las diferentes variables y métodos o ayudar a la detección de los errores señalando estos sobre el código, entre otros. Incorpora un sistema de depuración que permite compilar línea por línea las sentencias y permite establecer puntos de interrupción. En la siguiente imagen (Véase Figura 18) se pueden comprobar una captura de pantalla de QtCreator, entorno utilizado para el desarrollo del proyecto:

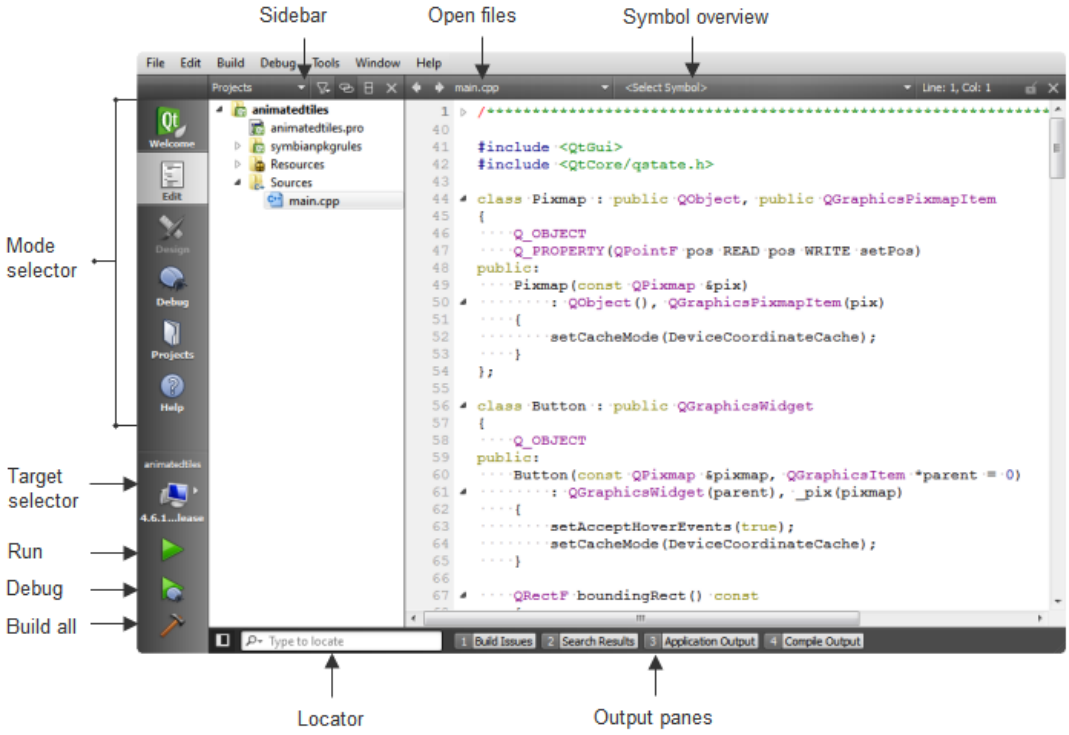


Figura 18. Entorno de programación QtCreator 2.3.

3.2. SENSOR KINECT

En este apartado, se hablará acerca de la cámara Kinect (Véase Figura 19). Se trata de un sensor creado por Primesense para Microsoft con el fin de ser incorporado en la videoconsola XBOX 360. Es capaz de percibir y reconocer los objetos a partir de dos cámaras frontales (una que capta datos RGB y un sensor de profundidad), así como un conjunto de 4 micrófonos. Para la percepción de profundidad se necesitarán 3 partes:

- Proyector de rayos infrarrojos.
- Sensor CMOS.
- Microchip que procesa la información.

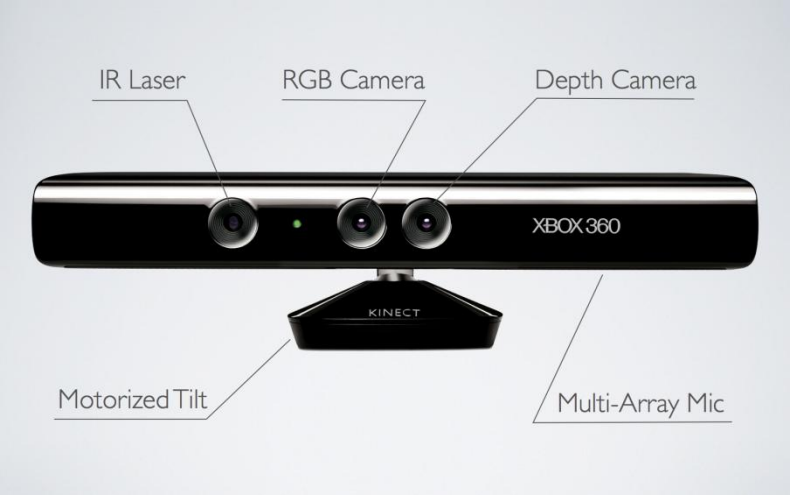


Figura 19. Partes de sensor Kinect para XBOX 360.

Este periférico será utilizado en el proyecto para obtener una representación de la realidad y formar las nubes de puntos que posteriormente serán tratadas para cumplir unos determinados objetivos. Para poder trabajar con este equipo, ROS necesitará *freenect_stack* [78] que incluye todos los topics de Kinect para poder acceder a la información del sensor.

3.2.1. Características principales y especificaciones técnicas

En este apartado se explicarán las características principales de cada uno de los componentes de Kinect, así como sus especificaciones técnicas. Posteriormente, se hará referencia al modo en el que este sensor capta los datos de profundidad y color. Los diferentes componentes de este hardware [79] son:

- Micrófonos: Conjunto de 4 micrófonos que nos permiten comunicarnos con el sistema o darle órdenes mediante el uso de la voz.
- Proyector láser: Emisor de rayos infrarrojos.
    - El láser se mantiene constante, es decir, no funciona por pulsos.
    - Longitud de onda: 830 nm.
    - Tiene un sistema de difracción que divide el láser en numerosos puntos aleatorios.
    - Potencia de salida: 60mW.
    - Contiene un estabilizador de temperatura, puesto que un pequeño cambio de esta puede modificar la longitud de onda.
- Sensor CMOS: Capta los rayos infrarrojos.
    - Tamaño de píxel: 5,2 µm.
    - 30 fps programables.
    - Resolución del sensor: 1,3 Megapíxeles.
    - Soporta entre 0-70 ºC de temperatura.
    - No puede medir temperaturas.
    - Campo de visión horizontal de 57º, mientras que el vertical es de 43,5º con una captación de profundidad comprendida entre 0,4 y 4 metros.
- Motor de inclinación: La cámara puede tener un rango de orientación de -27º a 27º.
- Cámara RGB: Se compone de un sensor CMOS de imagen con las siguientes características:
    - Tamaño de píxel: 2,8 µm x 2,8 µm.
    - 15 fps a máxima resolución y 30 fps a 640x512.
    - Resolución del sensor: 1289 H -1024 V, y 1,3 Megapíxeles.
    - Soporta entre -30-70 ºC de temperatura.
    - Filtro Bayer para captar el color.
    - Soporta VGA, QVGA, CIF y QCIF.
- Conector USB: Se trata de un USB 2.0 conectado al ordenador y a un transformador de CA/CC.
- Transformador de CA/CC: Este transformador se conecta al USB del sensor ya que la cámara necesita una mayor potencia para visualizar. Esto se debe a que el motor de inclinación acapara la mayoría de la potencia (Véase Figura 20).

.

Figura 20. Transformador para Kinect.

El funcionamiento del sensor Kinect se compone de dos partes:

- RGB: El sensor CMOS de imagen percibe los rayos luminosos que chocan contra los objetos y crea una representación bidimensional en color.
- Profundidad: Aquí es donde aparece el misterio de cómo puede captar Kinect datos de profundidad. El emisor lanza un láser que se dispersa formando un patrón de puntos aleatorios, de tal forma que, cuando aparece alguna variación en este patrón, significa que se ha interpuesto un objeto y se podrá extraer tanto su posición como su forma a partir de un proceso de triangulación realizado por el sensor CMOS. Es importante tener en cuenta que para poder llevar a cabo la triangulación es necesario un patrón de referencia. En la siguiente imagen (Véase Figura 21), se puede observar un patrón de puntos sobre una pared blanca, donde cabe destacar que existen una gran cantidad de puntos que no siguen un orden fijo, es decir, son puntos aleatorios y que parece que reflejan una matriz de 3x3 sobre la superficie. [80]

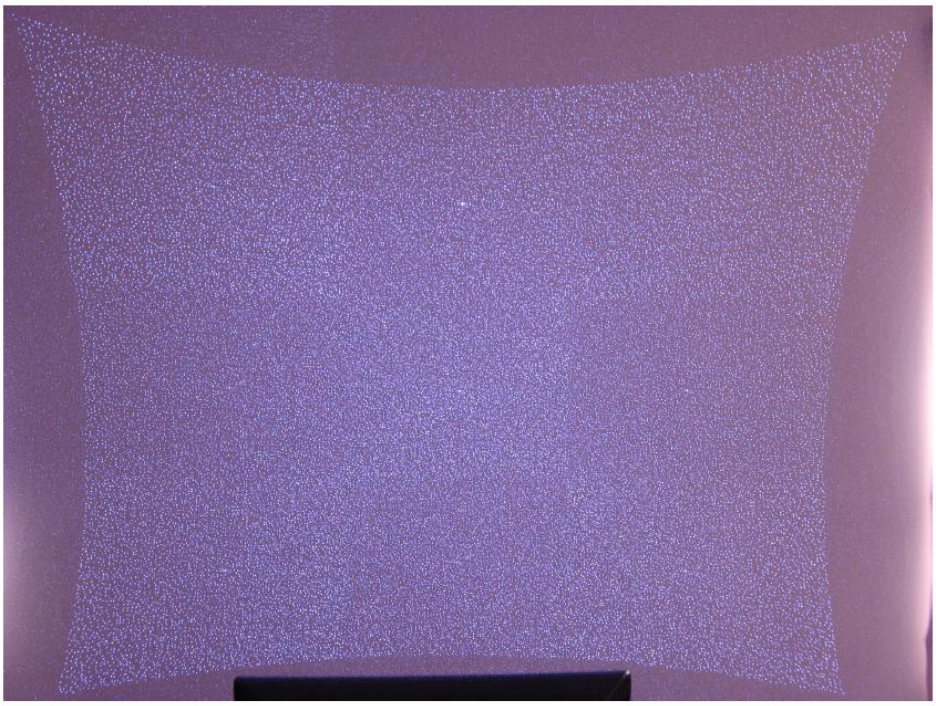


Figura 21. Patrón de puntos sobre una pared blanca.

Con esto lo que se quiere demostrar es que cada sensor Kinect tiene su propio patrón de referencia, el cual adquiere desde la fabricación y que se trata de un plano precargado en la memoria a una cierta distancia del sensor. Es por esto que cuando un objeto se interpone en el camino de los rayos, se produce un desplazamiento de los puntos sobre la superficie del objeto que distorsionan la señal (Véase Figura 22).

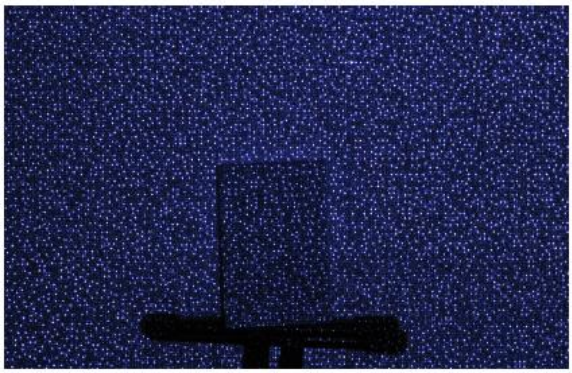


Figura 22. Distorsión del patrón al introducir un objeto en el entorno.

Este desplazamiento permite al sensor CMOS establecer una relación triangular que determina la distancia de la superficie captada con respecto al plano de referencia, extrayendo de esta forma la profundidad.

3.2.2. Calibración

Existen diferentes causas que pueden afectar a la correcta visualización de la cámara, como por ejemplo el valor del coeficiente de distorsión de la lente, el ángulo de captura o la posición del sensor, entre otros [81]. En la siguiente imagen (Véase Figura 23) se puede ver cómo Kinect pierde información de color o profundidad al sobrepasar los ángulos de visualización de cada cámara:

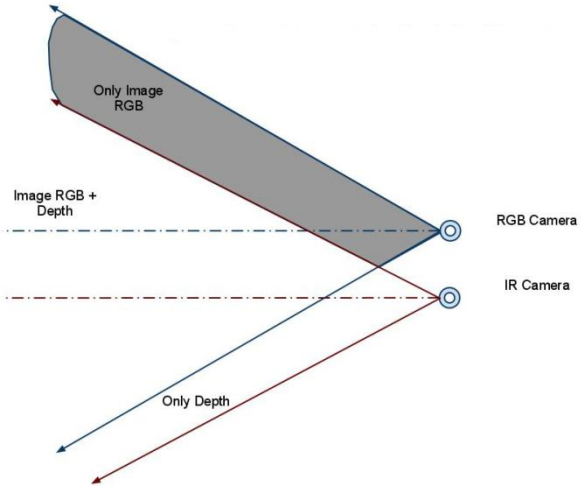


Figura 23. Funcionamiento de Kinect.

Por este motivo, es necesario realizar una calibración en dos niveles, para reducir el efecto de estos parámetros en el funcionamiento del sistema:

- Calibración intrínseca: Enfocada a los propios parámetros del conjunto de cámara y óptica: distancia focal, tamaño de los píxeles, etc. Para llevar a cabo esta calibración, se necesita la herramienta *camera_calibration* [82] (Veáse Figura 24) y un tablero de ajedrez para que la señal reconozca las intersecciones entre ellos.

Figura 24. Herramienta *camera_calibration*.

Repetiremos este método dos veces: uno para la cámara RGB y otro para la cámara de detección de profundidad (tapando el emisor de rayos infrarrojos, Véase Figura 25).

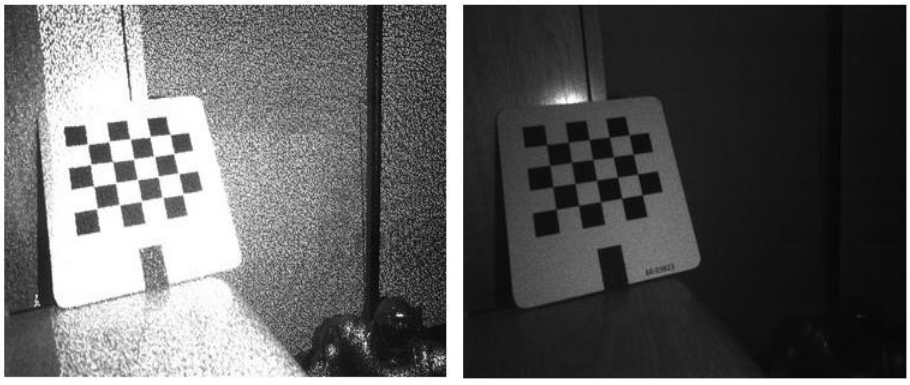


Figura 25. Tablero sin tapar y tapando el emisor infrarrojo, respectivamente.

Para cada cámara (color y profundidad), se deberán realizar varios movimientos (Véase Figura 26), alrededor de los diferentes ejes de coordenadas, asegurando que se utiliza todo el espacio visual del sensor.

Figura 26. Movimientos del tablero.

- Calibración extrínseca: Se refiere a la orientación y posición de la cámara. Además, permite hacer un ajuste de profundidad y color con el fin de visualizar la nube de puntos sin perder información. Como para la versión de ROS Hydro no se ha creado ningún método para solucionar este problema, esta calibración se realizó modificando los valores de los offset de forma manual y estimativa mediante la utilización de la herramienta *rqt_reconfigure* [83] (Véase Figura 27).

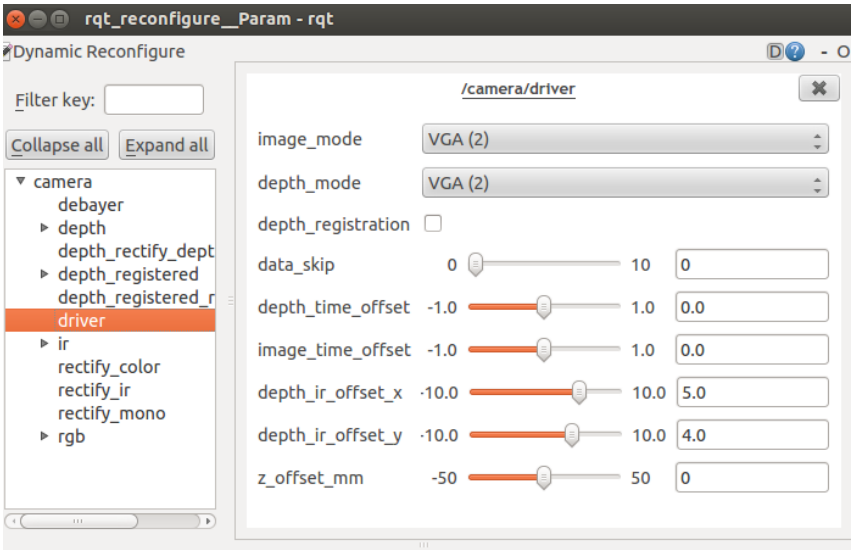


Figura 27. Herramienta *rqt_reconfigure*.

Cabe destacar que la señal que posteriormente se capta con Kinect en el proyecto pasa por un proceso de correlación de los puntos para alinear la nube, por lo que realmente esta calibración es importante para que no influya en la solución final. No obstante, *freenect_stack* [78] incorpora patrones que hacen que prácticamente no sea necesario tener que calibrarla.

Por lo tanto, se puede concluir que la calibración de esta cámara ayuda en gran medida a una mejor percepción de la realidad para evitar mediciones erróneas o representaciones desfasadas. Este proceso es obligatorio para cámaras en 3D y, de esta forma, asegurar una buena correlación de todos sus puntos en el espacio.

3.2.3. Kinect vs cámaras convencionales

En este apartado, se observarán las peculiaridades del sensor Kinect, que le diferencian del resto de las cámaras convencionales:

- Sensor Kinect:
  - Contiene dos cámaras (una de profundidad y otra de datos RGB) y un emisor de rayos infrarrojos. Permite captar la realidad de forma tridimensional al poder percibir la profundidad de los objetos mediante la emisión de rayos infrarrojos.
  - Solamente percibe los datos de profundidad de forma correcta en interiores puesto que el choque de los rayos luminosos con los rayos infrarrojos pueden distorsionar la señal. Los datos de color pueden extraerse perfectamente tanto en interiores como en exteriores.
  - Aunque obtiene mejores resultados con buena luminosidad, también puede usarse en entornos con mala visibilidad (por ejemplo, por la noche). Una iluminación extrema, puede alterar la recepción de los datos de profundidad.
  - Usabilidad para el desarrollo de aplicaciones en tiempo real, puede ser utilizado en varios sistemas operativos (Ubuntu, Windows, MAC) y permite visualizar los objetos de varias formas (en color, escala de grises, profundidad, etc)
  - Necesidad de cableado y conexión a la red eléctrica mediante un transformador de corriente alterna a corriente continua.
  - También utilizado con fines recreativos (XBOX 360).
- Cámara convencional:
  - Una sola cámara que capta datos en RGB.
  - Puede ser utilizado tanto en interiores como en exteriores.
  - Solo obtiene resultados visibles en condiciones de buena iluminación.

- Necesidad de batería o pilas para su funcionamiento (dependiendo del equipo).
- Solo visualiza en color, mayor dificultad para desarrollar aplicaciones en tiempo real (a no ser que sean mediante Webcam), puede ser utilizado en varios sistemas operativos. Con frecuencia, este tipo de cámaras captan señales (imágenes o videos), que serán tratadas por computador posteriormente.
- No puede ser usado para fines recreativos.

Diseño de aplicación basada en Kinect para vehículo inteligente

Capítulo 4: Diseño y desarrollo del proyecto

4.1. Planteamiento del problema

En la actualidad, la falta de atención al volante supone aproximadamente el 50% de los accidentes mortales en las carreteras españolas (sin datos de Cataluña), por encima de otras causas como pueden ser el incumplimiento de las normas de tráfico, la velocidad inapropiada y el cansancio o sueño en el conductor. Según datos del 2012, facilitados por el Real Automóvil Club de España (RACE), junto a BP y CASTROL, estas cifras serían de alrededor 517 personas fallecidas. Los principales motivos de estos sucesos son: salidas de vía, choques de vehículos producidos por alcances y atropellos. [84]

Se habla de *distracciones propias* cuando el conductor se encuentra atento a otras ocupaciones distintas a la conducción, y de *distracciones ajenas* cuando este fenómeno lo observamos en los automovilistas de nuestro alrededor. [84]

Las distracciones más casuales [85] que provocan este tipo de accidentes son:

- Ocupantes y mascotas: Ocurre al haber discusiones entre los diferentes individuos del coche, aunque el caso más significativo es cuando se viaja con niños y mascotas al tener que observar y controlar continuamente sus movimientos en los asientos traseros.
- Preocupaciones y estado de ánimo: Momentos de estrés, intranquilidad, nerviosismo, preocupaciones personales o conducir con prisas pueden favorecer la aparición de situaciones peligrosas.
- Dispositivos del coche: El hecho de soltar el volante para utilizar un dispositivo electrónico (por ejemplo, navegadores, calefacción, radio o dispositivos móviles) en el interior del coche es otra de las causas a tener en cuenta. Haremos especial referencia al uso del teléfono móvil, el cual produce distracciones al comunicarse con él, incluso con manos libres, o chatear.
- Apartar la vista del tráfico: La velocidad y el tiempo que trascurre sin poner atención a la vía, juegan un papel importante puesto que la distancia que recorreremos será superior cuando estos parámetros sean mayores. Principalmente, se producen al fijar el interés en el paisaje, carteles, otros accidentes, etc.
- Otros factores: Pueden darse casos en los que diversos factores, internos o externos, favorezcan la aparición de accidentes por esta razón. Nos referimos a temas como el cansancio, consumo de alcohol o estupefacientes, condiciones climáticas, atascos, señalización incorrecta o malas condiciones de iluminación, entre otros. Cabe destacar que existen también muchos casos en los que las propias distracciones son debidas a acciones personales de nuestra vida cotidiana como pueden ser asearnos, fumar o comer dentro del vehículo.

Para este proyecto, se ha querido poner especial énfasis en las distracciones provocadas por la falta de observación en la carretera o por el sueño. Para ello, hemos pensado en utilizar un sensor no muy costoso (Kinect) que vigile al conductor y realice una buena captación de la situación. Esto supondrá una ayuda para detectar este tipo de casos e intentar evitarlos.

4.2. Desarrollo del proyecto

Una vez conocido el problema a solucionar y las herramientas que se utilizarán, se procederá a explicar de forma clara y ordenada el desarrollo llevado a cabo para crear la aplicación.

Realizar una vigilancia del conductor es importante para comprender sus comportamientos, errores o necesidades. Para desempeñar una correcta monitorización del conductor, se pensó en hacer un seguimiento en 3D, que asegure no perder información de las facciones de la cara. Actualmente, el sensor Kinect es una herramienta de bajo coste que se ajusta a las necesidades del proyecto al facilitar la posibilidad de poder fusionar profundidad y color, por lo tanto, puede aportar resultados fiables.

El proyecto se divide en 3 partes, las cuales se corresponden con cada uno de los nodos o ejecutables del sistema: adquisición de la nube de puntos mediante sensor Kinect, detección de la orientación de la cara y visualización de los resultados. Todos estos nodos se encuentran perfectamente comunicados entre sí, realizando cada uno de ellos una tarea diferente dentro de la plataforma.

4.2.1. Adquisición de la nube de puntos

En este apartado, se analizará cómo se comporta el primero de los nodos, el cual se encarga de adquirir la nube de puntos. Para ello, es necesario subscribirnos a 5 topics, que servirán para obtener la señal 3D y que publicar el mapa final. Este ejecutable se basará en sincronizar los diferentes temas en tiempo real y extraer los datos de profundidad y color del entorno para posteriormente formar la nube de puntos.

4.2.1.1. Sincronización en tiempo real

Para poder recibir e interpretar correctamente la señal se deberá trabajar con una nube de puntos óptima. Por este motivo, es necesario acceder simultáneamente a los datos que recogemos del sensor para evitar un desfase de tiempo entre ellos que pueda suponer un problema para el desarrollo de la aplicación. Cuando se produce esta disparidad, es imposible que la nube de puntos se pueda analizar en tiempo real, puesto que nos subscribimos al color y la profundidad por separado.

Para evitar este problema, se han creado unas funciones que nos indican cuando se recibe la información de los topics de Kinect. Su objetivo es permitir trabajar con las muestras extraídas si se acceden a ellas al mismo tiempo (Véase Figura 28). De esta forma, se evita el desfase y se obtienen resultados de color y profundidad en tiempo real, por lo tanto, se puede realizar un estudio más exacto del entorno.

Figura 28. Color y profundidad en tiempo real.

Después de obtener toda esta información, se volcará a una única nube de puntos del tipo *pcl::PointCloud<pcl::PointXYZRGBA>* [86]. En ella, se almacenarán todos los datos y se irán actualizando para estudiar el entorno en 3D en todo momento.

4.2.1.2. Adquisición de datos de profundidad

Tras calibrar la cámara Kinect, nos subscribiremos al topic */camera/depth/points* para acceder a los datos de profundidad, es decir, se extraerá la realidad captada por los rayos infrarrojos del sensor.

Esta señal será tratada como una variable del tipo *sensor_msgs/PointCloud2* [87], que son los mensajes utilizados por ROS para referirse a una nube de puntos. Primeramente, hay que convertirlos a formato de Point Cloud Library para poder trabajar y manipular la muestra con un tamaño original de 488x640 píxeles. Posteriormente, se atribuyen estas mismas dimensiones al mapa final para volcar toda la información del topic sobre ella (Véase Figura 29).

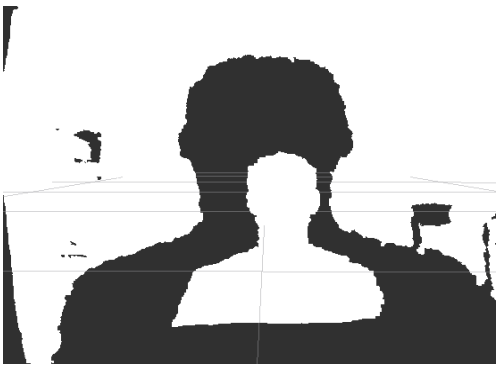


Figura 29. Nube de puntos profundidad.

Al mismo tiempo, se estarán lanzando por otro lado los temas referentes al color, con el fin de correlacionar todos los puntos XYZRGBA en un único modelo en 3 dimensiones.

4.2.1.3. Adquisición de datos de color

Para captar el color de las imágenes se necesita la interacción de dos topics: */camera/rgb/image_color* y */camera/rgb/camera_info*. El primero de ellos extrae los datos RGBA del entorno, mientras que el segundo percibe la información relativa a la imagen. Este último será importante a la hora de realizar el *Modelo Pin-Hole* [11] que posteriormente se explicará con mayor detalle.

El primer tema, percibirá una señal que se tratará como una variable del tipo *sensor_msgs/Image* [88], que es el mensaje que usa ROS para hablar de una imagen. Nada más subscribirnos a este topic, se deberá convertir esta señal a formato OpenCV para poder acceder a la imagen. Para ello, será imprescindible utilizar una herramienta de ROS llamada *CvBridge* [74] (Véase Figura 30) que permitirá realizar la conversión directamente en el formato que se quiera (en nuestro caso, BGRA8 [89] para captar también datos de transparencia). El tamaño con el que son percibidas estas muestras es de 480x640 píxeles, pero para que

coincidan con el alto y ancho de las nubes de puntos se han tenido que reajustar estos parámetros para que se trabaje en su totalidad con señales de 488x640 píxeles.

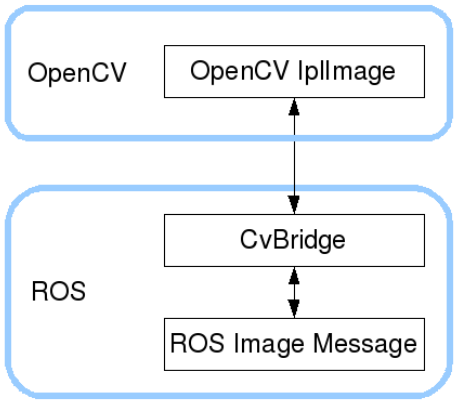


Figura 30. Esquema de conversión ROS-OpenCV.

Una vez extraída la imagen (Véase Figura 31), se procederá a transformar esta señal bidimensional en una señal tridimensional, para poder atribuir a cada píxel su correspondiente valor de profundidad. Por este motivo, es necesario que los dos topics (color y profundidad) se ejecuten al mismo tiempo, para que esta correlación se desarrolle de forma satisfactoria.

Figura 31. Imagen en color.

4.2.1.3.1. Modelo Pin-Hole

El motivo por el que hay que subscribirse al topic que proporciona la información de la imagen, es para poder desarrollar el *Modelo Pin-Hole*. Primeramente, es necesario conocer el concepto de *distancia focal*, la cual se define

como la distancia existente entre el eje de la lente de la cámara y el punto del objeto a percibir. [90]

Esta teoría consiste en únicamente darle importancia a aquel punto captado por el rayo luminoso que se refleja sobre un objeto y pasa por la distancia focal (Véase Figura 32), rechazando aquellos que no cumplen este requisito.

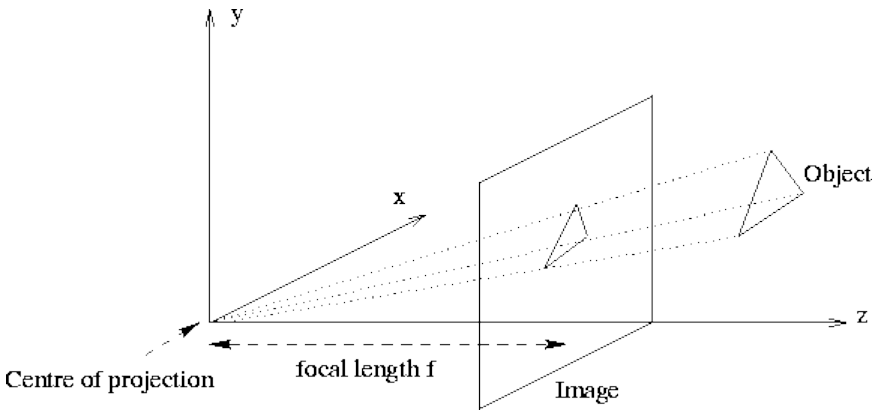


Figura 32. Modelo Pin-Hole.

Teniendo en cuenta las coordenadas del espacio (X, Y, Z) y de la imagen (x_{imagen}, y_{imagen}) se puede establecer una relación entre ellas. Para ello, es imprescindible conocer la profundidad (Z) para cada punto. Este valor es conocido puesto que Kinect mide en este eje. Estos datos serán calculados por semejanza de triángulos de la siguiente forma:

(1) $$x_{imagen} = \frac{distancia\ focal}{Z} \cdot X$$

(2) $$y_{imagen} = \frac{distancia\ focal}{Z} \cdot Y$$

Solamente se tiene conocimiento sobre las columnas y filas de las que se compone la imagen, pero no se tiene información acerca de las coordenadas de cada píxel (x_{imagen}, y_{imagen}). Por lo tanto, habrá que extraer las proyecciones vertical y horizontal, que permitirán acceder a estos datos para poder atribuir a cada punto del espacio su valor de color específico. Para ello la librería OpenCV, mediante una sentencia, relaciona directamente el vector formado por las coordenadas (X, Y, Z) de cada punto de la realidad con sus proyecciones en la imagen. A partir de esas proyecciones, esta biblioteca calcula la coordenada de cada píxel en la imagen. Una vez conocidas, se pueden asignar los datos RGB a cada punto del espacio, obteniendo la nube de puntos en color.

4.2.1.4. Correlación de datos de profundidad y color

Una vez obtenidos los datos de profundidad y color en tiempo real, y tras aplicar el *Modelo Pin-Hole*, nos queda un último paso para formar la nube de puntos. Este consiste en mezclar toda la información, asignando a cada valor de profundidad su correspondiente color. Para ello, habrá que recorrer el mapa e ir enlazándolos de forma individual para cada píxel, es decir, punto por punto. No obstante, cuando se realiza esta correlación, los puntos pueden que se encuentren desalineados como ocurre en el caso de este proyecto (Véase Figura 33). Esto es debido en gran medida al ángulo del sensor y a su posición, por lo tanto, se deberá corregir este fenómeno para que no afecte al funcionamiento de la aplicación.

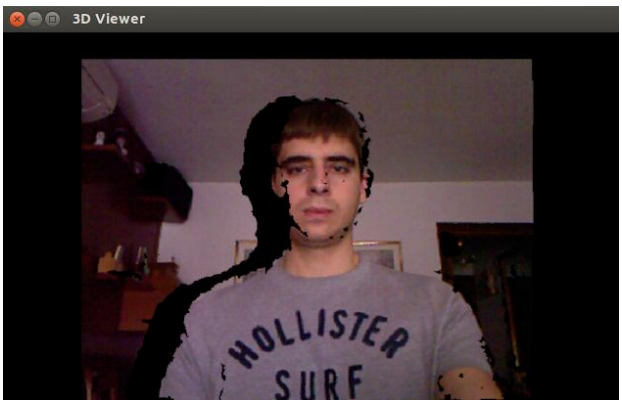


Figura 33. Nube de puntos en color desalineada.

Como se puede observar en esta imagen, los datos no se encuentran perfectamente alineados puesto que la profundidad está ligeramente recostada hacia la derecha y arriba del color.

Por este motivo, se han creado dos nuevos topics del tipo *sensor_msgs/Int16*, uno que indicará el valor del offset para el eje X y otro para el eje Y. No obstante, para poder utilizarlos primero tenemos que convertir este formato en un entero para poder trabajar con él. Estos datos se sumarán a las posiciones extraídas del *Modelo Pin-Hole* con el fin de desplazar la imagen de arriba hacia abajo o lateralmente para asegurar una correlación óptima. No será necesario aplicarlo en Z puesto que Kinect capta los datos a partir de este eje, por consiguiente ya estará alineado en esta dirección. Tras probar con varias cifras, se ha llegado a la conclusión de que el desfase a introducir es -2 píxeles para el eje X y 8 para el eje Y. Los resultados obtenidos a partir de este arreglo son fiables y satisfactorios, obteniendo una nube de puntos final como la que observamos en la siguiente captura (Véase Figura 34) del visualizador:

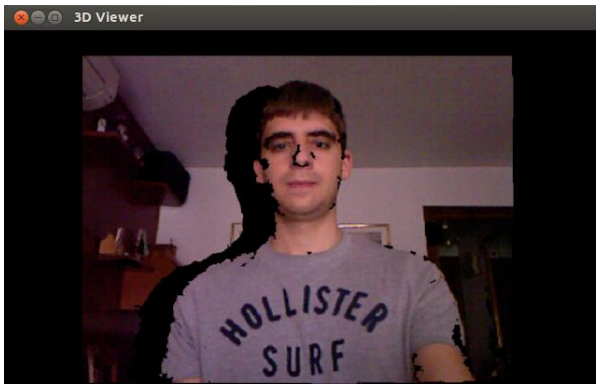


Figura 34. Nube de puntos en color alineada.

El único inconveniente de este ajuste, es que se pierde la información del entorno que queda fuera de la nube de puntos, por lo que se añadirán píxeles en negro para compensarlo y proceder a su posterior análisis. Para asegurar una mejor comprensión, se explicará este fenómeno paso por paso:

- En primer lugar, se añadirá el offset en X con un valor de -2 píxeles por fila. Con esto, se estará desplazando el color hacia la derecha, por lo que habrán 2 píxeles por fila que saldrán de la nube de puntos y otros 2 que se añadirán en la parte izquierda de ésta, los cuales se apreciarán como puntos en negro para que no intercedan en los resultados.
- De igual forma, se establecerá como 8 píxeles por columna el valor del desfase en el eje Y. En este caso, los datos RGBA se estarán desplazando hacia arriba en el mapa, perdiendo las mediciones de los 8 píxeles por columna que salen fuera del mapa y añadiendo otros 8 en negro al final de cada una de ellas.

Aplicando este método, se está recortando el espacio de trabajo pero se obtienen resultados más fiables y exactos. Por este motivo, lo ideal sería que la cantidad de puntos a desplazar no sean de un valor absoluto muy alto puesto que estaríamos limitando la realidad a estudiar.

Por último, comentar que a estos topics se accederán mediante una interfaz llamada *rqt_gui* [91] (Véase Figura 35), la cual proporciona ROS para poder trabajar con ellos cuando sea necesario. También, se puede especificar la frecuencia con la que queremos que se modifiiquen estas cifras, las cuales han sido mantenidas a 1 Hz puesto que se aplicarán en todo momento y de forma constante como el resto de temas de la aplicación.

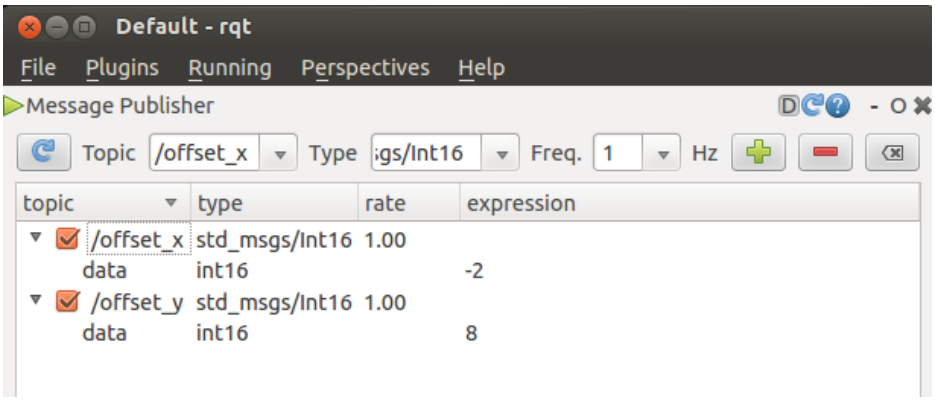


Figura 35. Interfaz *rqt_gui* con los offsets activos.

4.2.2. Detección de la orientación de la cara

Una vez extraído el mapa a color y ajustado el offset, se procederá a captar la información correspondiente a la cabeza para realizar el posterior análisis facial. Para ello, se realizará una estimación de la posición de la cabeza con 3 grados de libertad (ángulos de Euler [92]) utilizando el algoritmo ICP (Iterative Closest Point) [19]. Esta parte del proyecto, consistirá en identificar las diferentes regiones de la cara, seguido de una detección y separación del entorno, y finalmente se obtendrá la orientación de ésta haciendo un estudio exhaustivo de su comportamiento para cada movimiento (Véase Figura 36).

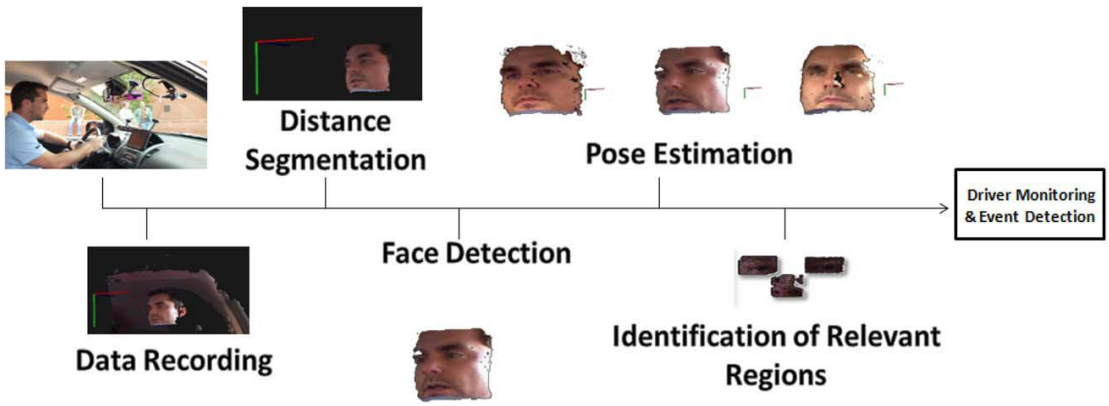


Figura 36. Esquema del comportamiento de la aplicación.

El objetivo final ha sido crear una aplicación que monitorice la cabeza en 3D, utilizando un sensor de bajo coste (Kinect) que puede resultar muy útil a la hora de investigar diversos factores humanos, poniendo especial énfasis en la vigilancia del

conductor para captar las distracciones al volante (detección de rostro, falta de atención o sueño, entre otros).

4.2.2.1. Detección facial y segmentación

Para realizar una buena detección de la cara (Véase Figura 37), la ubicación del sensor ha sido elegida en base a algunas limitaciones [3]:

- Distancia entre sensor y conductor mínima: 0,5 m.
- Kinect debe tener siempre el sujeto a la vista, por lo tanto, la cabeza del individuo se tiene que apreciar perfectamente. Este aspecto es importante puesto que el volante o los movimientos que se realizarán al maniobrar, no tendrán que interferir en el sistema. No obstante, la orientación del sensor se ha ajustado para que no ocurran este tipo de problemas.
- La posición de la cámara será lo más baja posible para evitar quitar visibilidad de la carretera al conductor. Su colocación no deberá afectar al campo visual del automovilista.

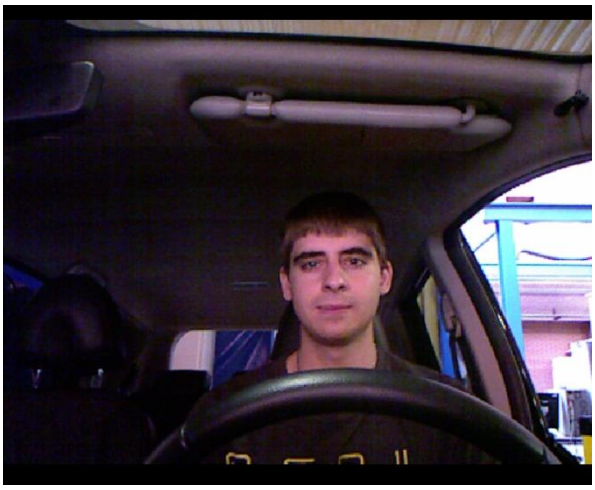


Figura 37. Vista del conductor en el vehículo inteligente.

Tras adquirir la nube de puntos alineada en color, se aplicará un filtro distancia para quitar el fondo con el fin de reducir los puntos a analizar, aumentar la velocidad de procesamiento y limitar el espacio de búsqueda. Una vez filtrada, se obtendrá de ella una imagen en color empleando un modelo de proyección de los puntos y se extraerán los datos RGB (Rojo, Verde, Azul). Esto se realiza para la posterior detección de la cabeza mediante el algoritmo de *Viola-Jones* [24]. Tras este último paso, se construirá un nuevo mapa 3D con los resultados faciales, mediante la extracción de los puntos que se encuentran en el rectángulo donde está

la cara detectada y su información correspondiente de color. Para restringir aún más el modelo final, se llevará a cabo una erosión (recortar la señal) de un 15% en los límites superior e inferior de este rectángulo, eliminando cuello y pelo ya que pueden proporcionar datos erróneos, y permitiendo estudiar solamente la información referente a la cara (Véase Figura 38). [3]

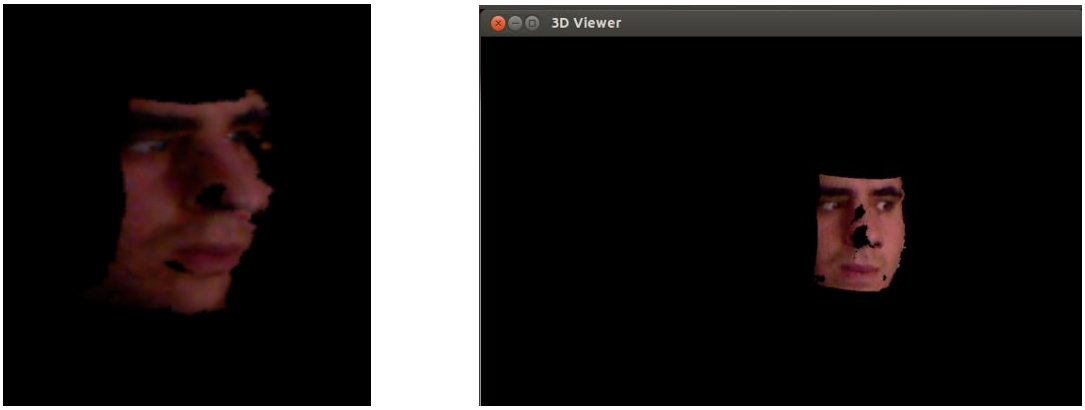


Figura 38. Detección de la cara.

4.2.2.2. Identificación de los ángulos de la cara

Para reducir la cantidad de puntos se utiliza un filtro VoxelGrid, el cual consiste en dividir la nube de puntos en pequeñas cajas 3D (*voxel*) en el espacio. Posteriormente, para cada *voxel* se aproximan todos sus puntos hacia el centro de gravedad, también llamado centroide (Véase Figura 39) [20]. Cuanto mayor sea el parámetro de este filtro, se contará con menos puntos, se minimizará en mayor medida la resolución y se ganará velocidad de procesamiento. Cabe destacar que esto se aplica tanto a la nube que se toma de referencia como al mapa actual.

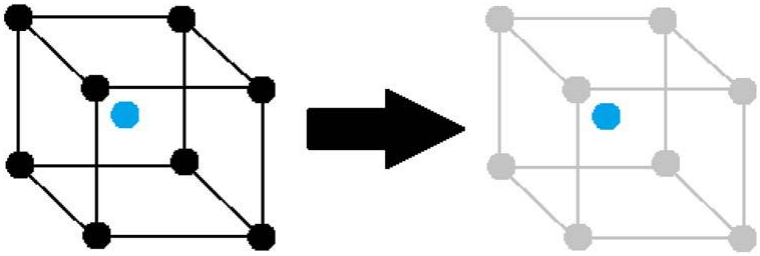


Figura 39. Filtro *VoxelGrid*.

Una vez definidos, se aplicará el algoritmo ICP (Iterative Closest Point), el cual se utiliza para reducir al mínimo las diferencias entre la señal actual y la referencia. Consiste en que un punto de la referencia se mantiene fijo, mientras que

en la representación actual, se produce una transformación mediante la combinación de una matriz de rotación y un vector de traslación para minimizar la distancia existente entre el punto de la nube actual y el de la referencia [19]. Las fases de este proceso [3] son:

1. Primeramente, los puntos que se encuentren más próximos entre las nubes de puntos se relacionan para proceder a su comparación, donde d es la distancia entre los puntos p_1 (perteneciente al modelo) y p_2 (referente al mapa a estudiar):

   (3) $$d(p_1, p_2) = \|p_1 - p_2\|$$

2. Seguidamente, se calculará los parámetros que se utilizarán para la transformación (rotación y traslación) por el método de *Mínimos Cuadrados* [93]:

   (4) $$MC = \frac{1}{n}\sum_{i=1}^{n}(p_1 - p_2)^2$$

3. El paso posterior es crear la matriz de transformación teniendo en cuenta el vector de traslación y que la rotación está formada por una matriz para cada eje de coordenadas:
   a. Traslación:

   (5) $$T = [x_t \quad y_t \quad z_t]^T$$

   b. Rotación:

   (6) $$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Delta\varphi) & -sen(\Delta\varphi) \\ 0 & sen(\Delta\varphi) & \cos(\Delta\varphi) \end{bmatrix}$$

   (7) $$R_y = \begin{bmatrix} cos(\Delta\delta) & 0 & sen(\Delta\delta) \\ 0 & 1 & 0 \\ -sen(\Delta\delta) & 0 & cos(\Delta\delta) \end{bmatrix}$$

   (8) $$R_z = \begin{bmatrix} cos(\Delta\theta) & -sen(\Delta\theta) & 0 \\ sen(\Delta\theta) & cos(\Delta\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

   c. Matriz de transformación:

   (9) $$M_t = R\left(\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + T\right)$$

4. Se extraen los errores de entrenamiento entre la nube actual transformada y el mapa de referencia.
5. Posteriormente, se repiten todos estos pasos hasta que el error de entrenamiento es menor que el error mínimo especificado o el número de iteraciones alcance el número máximo definido. N es la cantidad de puntos que contienen los mapas, m_i el punto de la nube modelo, f es el error de entrenamiento, es decir, la comparación entre los *Mínimos cuadrados* de los puntos de la primera nube captada (m_i) y la última (t_i), y M_t la matriz de transformación:

(10) $$f(R,T) = \frac{1}{N}\sum_{i=1}^{n}\|m_i - M_t(t_i)\|^2$$

La finalidad de este algoritmo es sacar la matriz de transformación que minimice el error de entrenamiento y así poder extraer los ángulos de rotación de Euler. En todo momento, el efecto de traslación se elimina debido a que se trabaja con los centroides de las dos nubes para la obtención del rostro. Los grados de libertad son:

- Pitch: Movimiento de subida y bajada de la cabeza

(11) $$\Delta\theta = sen^{-1}(R_{y,z})$$

- Yaw: Movimiento lateral del rostro.

(12) $$\Delta\varphi = tan^{-1}\left(\frac{-R_{y,x}}{R_{y,y}}\right)$$

- Roll: Movimiento lateral oblicuo de la cara, de balanceo.

(13) $$\Delta\delta = tan^{-1}\left(\frac{-R_{x,z}}{R_{x,x}}\right)$$

4.2.3. Visualización

Este nodo servirá para visualizar los resultados, pero realmente no será necesario en el vehículo inteligente puesto que solamente se tiene interés en los ángulos de la cara. No obstante, este módulo es de gran utilidad para mostrar por pantalla el comportamiento de la aplicación y poder comprobar cómo varían los datos al realizar diferentes movimientos. Para ello, se tomará como referencia, es decir como eje de coordenadas, el centro de la cámara.

Para lanzar este módulo se necesitará la herramienta *PCLVisualizer* [94] que se puede extraer de la librería Point Cloud Library. Está creada con el fin de poder visualizar prototipos, así como mapas en 3D que nos ayuden a analizar los resultados finales tras aplicarle los algoritmos correspondientes. Aparte de estas funciones, también se pueden dibujar formas básicas en 3D (por ejemplo, esferas, conos o cilindros, entre otros) a partir de ecuaciones paramétricas. [95]

Este nodo, se subscribirá al topic publicado tras realizar la detección facial y la extracción de los ángulos y mostrará por pantalla la nube de puntos. Si en el entorno hay una cara, visualizará solo el rostro (Véase Figura 40), pero en caso contrario, se podrá observar la nube de puntos completa.

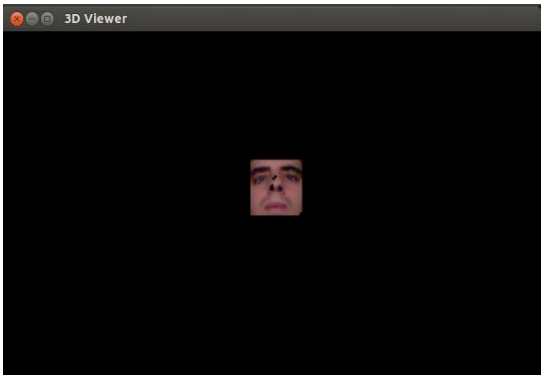


Figura 40. Visualizador cara.

Debido al sistema de referencia escogido, las representaciones se manifiestan con un desfase de 180º con respecto al eje Z. Por este motivo, se ha rotado π radianes la nube de puntos completa en torno a este eje con el fin de visualizarla correctamente. La siguiente ecuación [92] muestra la forma de resolver este problema, donde $\theta=\pi$, seguido de una captura de pantalla (Véase Figura 41) del antes y después de llevarla a cabo:

(14)
$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -sen(\theta) & 0 \\ sen(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

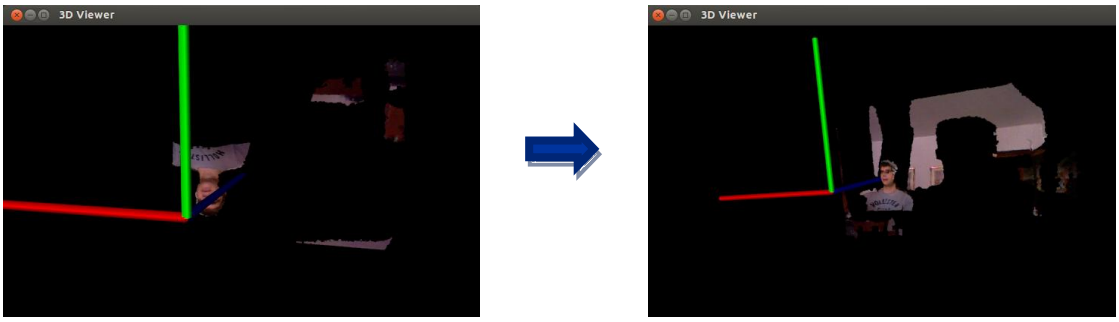


Figura 41. Antes y después de aplicar matriz de rotación.

Una vez rotada la señal, se podrá visualizar perfectamente el comportamiento del sistema. Este ejecutable, actualizará continuamente el mapa final, superponiendo una ventana del visualizador sobre otra, es decir, mantiene cada captura durante un segundo y posteriormente sobrescribe una nueva muestra sobre la que había, y así sucesivamente. Cabe destacar que, aunque la puesta en marcha de este módulo sea un poco lento, los datos son recogidos con una velocidad superior, puesto que funciona a 15-18 fps en el ordenador del alumno y aproximadamente a 30 fps en el computador del vehículo.

4.3. Ejecución de la aplicación

En este apartado, se comentarán los pasos a seguir para poner en marcha el proyecto. Cabe destacar que la ejecución de la plataforma se desempeñará desde el terminal de Ubuntu 12.04 LTS, sistema operativo en el que se ha trabajado. Antes de iniciar, lo primero es lanzar el Máster de ROS, para poder usar todas las herramientas de las que dispone este meta-sistema operativo. Seguidamente, se cargará el sensor Kinect mediante la herramienta *freenect_stack* [78] para permitir a la aplicación poder acceder a todos los topics de la cámara.

Una vez marcadas las pautas iniciales y se pueda trabajar con todas las herramientas de ROS y del sensor, se ejecutarán los 3 nodos en el siguiente orden:

1. *proyecto*: En este ejecutable se creará la nube de puntos en color alineada, en base a la información de Kinect y de los offsets.
2. *orientación*: En él se realizará la detección facial y la extracción de los ángulos de la cara que se encuentre en el mapa.
3. *visualizador*: Módulo que servirá para visualizar por pantalla las muestras extraídas y que ayudarán a comprender su funcionamiento para su futuro análisis.

Cuando ya estén todos los nodos funcionando, la señal aparecerá desalineada, por lo que habrá que subscribirse a los offset en X e Y para alinear la nube de puntos. Para ello, se lanzará la herramienta *rqt_gui* y se definirán los valores de los desfases en -2 para el eje horizontal y 8 para el vertical. Después de este último paso ya se puede estudiar detenidamente cómo se comporta este sistema y estudiar los resultados experimentales.

A modo de guía y para facilitar la comprensión de esta plataforma, se ha desarrollado un diagrama de flujo (Véase Figura 42) que explica de forma clara y sencilla las fases por las que iremos pasando tras su ejecución:

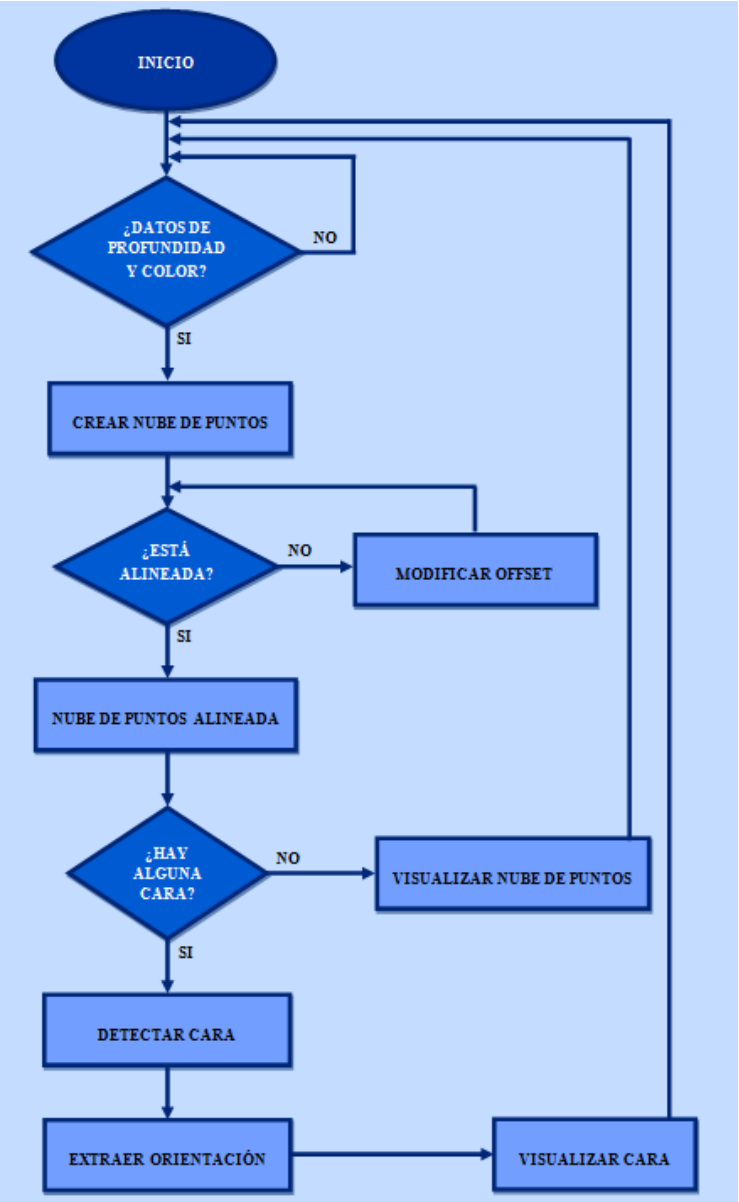


Figura 42. Diagrama de flujo del funcionamiento de la aplicación.

Primeramente, la aplicación recibirá la información de color y profundidad, las cuales deberán llegar simultáneamente. Si estos datos son capturados al mismo tiempo, se juntarán en una única nube de puntos con datos RGBA, en caso contrario no formará el mapa en color. Una vez tengamos una muestra, se analizará si esta se encuentra alineada o no, es decir, si existe una buena correlación entre profundidad y color. Dependiendo del ángulo y la posición del sensor habrá que asignar un valor diferente de offset, por lo tanto, hay que ir probando con diferentes cifras hasta observar que está perfectamente alineada. Posteriormente, se hará un estudio de la realidad captada, donde puede ocurrir lo siguiente:

- Haber cara: Se separarán sus facciones del resto del entorno, se extraerá su orientación en un fichero de texto y se visualizará únicamente el rostro.
- No haber cara: Se visualizará la nube de puntos alineada.

Finalmente, se volverá al punto inicial para realizar un examen completo y continuo de la situación repitiendo estos pasos para cada nube de puntos en tiempo real. En la siguiente imagen (Véase Figura 43), se puede apreciar un fragmento de la herramienta *rqt_graph* [96] que nos permite observar de forma gráfica lo que ocurre al poner en marcha la aplicación:

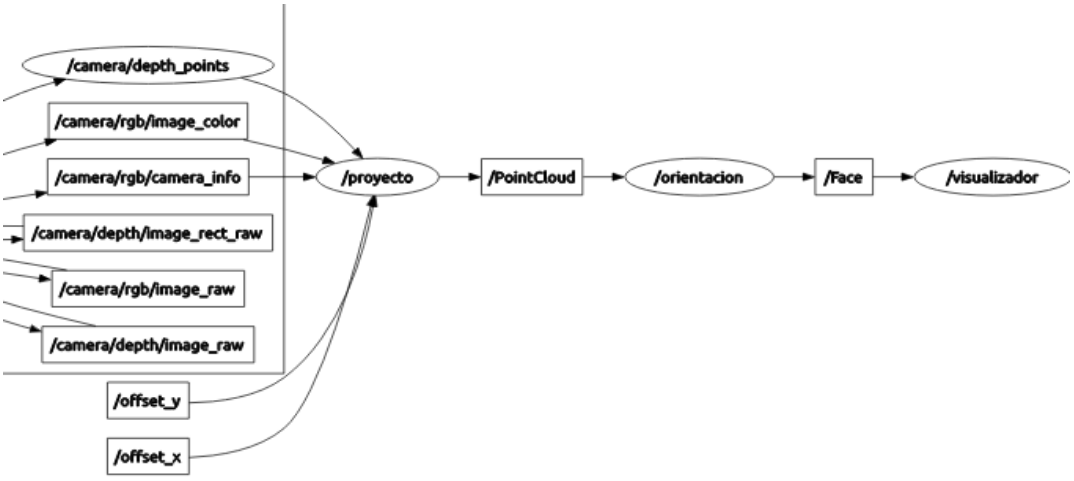


Figura 43. Fragmento de *rqt_graph* con la aplicación en marcha.

El nodo *proyecto* se subscribe a los topics de la cámara y a los offset para formar la nube de puntos alineada la cual publica como el tema *PointCloud*. De la misma forma, el nodo *orientación* adquiere esta información, detecta la cara y extrae los ángulos de esta, en caso de haberla, publicando los resultados en el topic *Face*. Finalmente, el nodo *visualizador*, se subscribirá a este topic y mostrará por pantalla la cara o el mapa en color, en caso de que exista o no exista cara en el entorno, respectivamente.

Para finalizar, añadir que, aunque no se aprecie en estas representaciones, se creó un topic llamado *orientación* del tipo *geometry_msgs/Point* [97] donde se irán publicando los valores de los ángulos de la cara. Realmente, su acometido es informativo, puesto que solamente nos servirá para apreciar los datos por pantalla, por lo que no afecta al funcionamiento de la plataforma.

Es importante añadir que la aplicación también se puso en marcha en el vehículo inteligente, pero para ello se utilizó ROS Indigo Igloo. Por este motivo, hubo que realizar pequeñas modificaciones en algún ejecutable debido a que se trata de una versión más actual de ROS. No obstante, el Departamento continúa adaptando el sistema para que funcione correctamente en esta última actualización, al haber modificaciones en las referencias que toma la herramienta para calcular los ángulos de Euler. Concretamente, se ve afectado cuando se realiza un movimiento Pitch hacia arriba.

Diseño de aplicación basada en Kinect para vehículo inteligente

Capítulo 5: Resultados prácticos

5.1. Análisis de los movimientos realizados

En este apartado, se estudiarán los movimientos que se van a realizar para obtener los resultados experimentales del proyecto, a partir de los cuales se extraerá la orientación de la cara. En todo momento, se analizará el entorno real sobre el mapa tridimensional en color, perfectamente alineado.

En primer lugar, comentar que se han realizado 4 pruebas para cada movimiento, adquiriendo los datos en diferentes franjas horarias (siempre con buena visibilidad y con luz natural). Dependiendo de este parámetro, se observará la nube de puntos en color con mejor o peor nitidez, pero mientras ésta se aprecie con claridad se obtendrán datos fiables.

Nada más lanzar la aplicación, el conductor deberá mantener durante escasos segundos la cabeza en posición frontal (Véase Figura 44) para que tome la referencia de la cara correctamente. Esto se deberá llevar a cabo siempre que se vuelva a iniciar la plataforma puesto que el sistema capta la orientación en base a esta posición inicial.

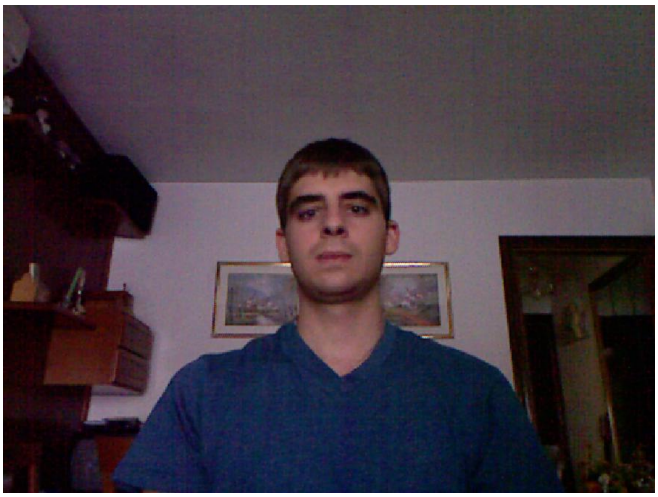


Figura 44. Posición inicial.

Los principales desplazamientos analizados son aquellos que permiten captar la forma en la que se va produciendo un cambio en los ángulos de cada eje de coordenadas. Por lo tanto, los resultados indicarán cuánto se ha movido la cara al rotar con respecto al eje correspondiente. Los principales movimientos a analizar son:

- Movimiento *Pitch*: Se trata de un desplazamiento alrededor del eje X, subiendo y bajando la cabeza. Las tres etapas (Véase Figura 45) por las que se pasará son:

    - Posición inicial.
    - Movimiento hacia arriba.
    - Movimiento hacia abajo.

Figura 45. Movimiento Pitch.

- Movimiento *Yaw*: Comúnmente denominado *movimiento lateral*. Consiste en hacer girar la cara con respecto al eje Y (Véase Figura 46). Las fases a llevar a cabo son:

    - Posición inicial.
    - Movimiento lateral hacia la derecha del conductor.
    - Movimiento lateral hacia la izquierda de la persona.

Figura 46. Movimiento Yaw.

- Movimiento *Roll*: Esta rotación consiste en variar el ángulo facial en torno al eje Z (Véase Figura 47). Este es el ángulo más importante a tener en cuenta en casos de somnolencia. Los pasos de los que consta son:

- Posición inicial.
- Movimiento lateral oblicuo hacia la derecha del individuo.
- Movimiento lateral oblicuo hacia la izquierda del automovilista.

Figura 47. Movimiento Roll.

5.2. Parámetros iniciales

A la hora de realizar una detección óptima de la cara y poder captar correctamente los datos, hay que tener en cuenta una serie de aspectos que pueden afectar al correcto funcionamiento de la aplicación. Por este motivo, antes de poner en marcha la herramienta, se han marcado unos parámetros iniciales a partir de los cuales se extrajeron todos los resultados del proyecto.

Principalmente, los dos parámetros más importantes a tener en cuenta serán la iluminación del entorno y la posición y orientación del sensor.

5.2.1. Iluminación

En cuanto a la iluminación, cabe destacar que los datos fueron tomados con luz natural. En todo momento, se ha querido simular un ambiente muy similar al que habrá dentro del vehículo, por lo tanto se ha ajustado el entorno lo máximo posible a situaciones reales.

Es importante recalcar que todas las pruebas realizadas se han llevado a cabo en condiciones de iluminación diferentes. Esto se pensó debido a que realmente este aspecto varía dependiendo de la hora del día (Véase Tabla 1), las condiciones meteorológicas, lunas o cristales tintados, etc. También, indicar que todos los resultados que aparecen en el proyecto fueron tomados con buena o baja visibilidad.

Prueba	Horario
1	09:00-10:00
2	13:00-14:00
3	17:00-18:00
4	21:00-22:00

Tabla 1. Horas de iluminación.

Por último, añadir las complicaciones del sistema en situaciones con exceso de iluminación (gran cantidad de reflejos) o de oscuridad dentro del vehículo (por ejemplo, de noche en carreteras no iluminadas o en túneles). En estos casos, las señales que capta el sensor Kinect pueden verse afectadas, dificultando una correcta correlación entre datos de profundidad y color que desembocan en mediciones erróneas.

5.2.2. Posición y orientación del sensor

Las pruebas han sido desarrolladas con la cámara Kinect para XBOX 360 cuyos parámetros iniciales (posición y orientación) se mantienen fijos durante la puesta en marcha de la aplicación. Estos dos aspectos son importantes puesto que al sufrir variaciones, pueden modificar el funcionamiento de la plataforma y obligarían a reajustar el sistema. Por este motivo, se ha pretendido simular una situación real donde todos los resultados han sido extraídos en un mismo entorno.

La cámara Kinect formaba un ángulo de 15º con respecto a la horizontal a una altura de 86 cm entre su base y el suelo. Esta se encontraba a 60-70 cm de distancia de la cara del conductor, la cual dista del suelo a unos 112 cm (Véase Figura 48).

Figura 48. Vista de Kinect y capturas lateral y frontal del conductor, respectivamente.

Cabe destacar que todas las pruebas han sido realizadas sobre la misma superficie puesto que en el vehículo inteligente será igual en todo momento. En la siguiente imagen (Véase Figura 49), se puede observar la nube de puntos ligeramente rotada junto con el eje de coordenadas (Eje X, rojo; Eje Y, verde; Eje Z, azul) que ha sido utilizado como referencia:

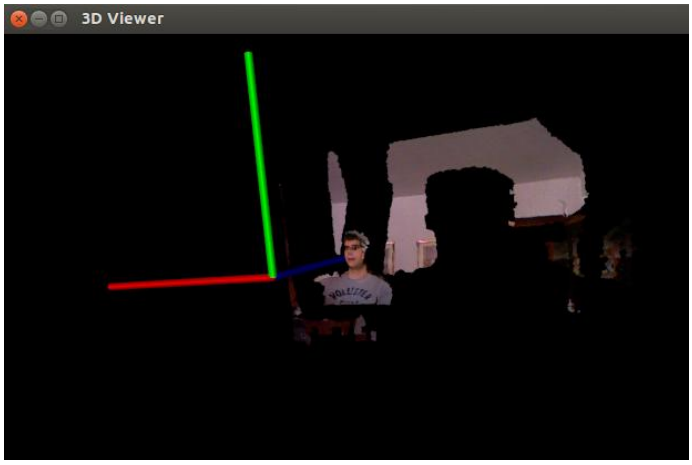


Figura 49. Nube de puntos ligeramente rotada y ejes de coordenadas.

Aparte de estos dos parámetros, también se deberá hacer referencia a que la nube de puntos no se encuentra totalmente alineada al recibir las señales de las cámaras de profundidad y color. Por lo tanto, se aplicará una serie de offset en los ejes horizontal y vertical, cuyo ajuste permite obtener datos más precisos (Véase Tabla 2).

Eje	Desplazamiento
X	-2 píxeles
Y	8 píxeles
Z	0 píxeles

Tabla 2. Offset en ejes de coordenadas.

Por último, se ha analizado el comportamiento modificando estos parámetros iniciales simulando que el individuo fuera de una estatura superior o inferior. Es importante recalcar que la aplicación funciona correctamente, pero se ha apreciado que, al tener valores nuevos de posición y orientación, hay que volver a adaptar los desfases en los ejes para alinear la nube de puntos.

5.3. Análisis de los resultados

Tras haber definido los movimientos y parámetros iniciales del entorno a estudiar y del sensor Kinect, se procede a poner en marcha el sistema. En todo momento, se extraen los datos en ficheros de texto y se obtienen resultados diferentes dependiendo del movimiento realizado. Posteriormente, se procede a analizar su comportamiento mediante el uso del visualizador, tablas y gráficas. Dependiendo del movimiento, habrá que fijarse directamente en un determinado ángulo, el cual nos servirá para determinar la posición de la cara en el momento de la rotación.

Para comprender su funcionamiento, se han realizado unos videos que muestran cómo capta la señal y trabaja con ella para alcanzar los objetivos del proyecto en cada uno de los movimientos: Pitch [98], Yaw [99] y Roll [100].

Hay que destacar que la velocidad con la que percibe los datos el sensor varía dependiendo de donde se esté ejecutando ya que en el ordenador donde se obtuvieron los datos se alcanzó una velocidad máxima de entre 15-18 fps, mientras que con el computador del coche (Véase Figura 50) llegaba a los 30 fps. Por este motivo, en el vehículo inteligente la ejecución de la aplicación sería bastante más rápida de lo que se aprecia en los videos.

Figura 50. Ordenador del vehículo y su ubicación.

5.3.1. Movimiento Pitch

Tras poner en marcha la aplicación y ajustar los offset, se procederá a analizar el movimiento descrito alrededor del eje X. En las siguientes capturas (Véase Figura 51), se podrá observar la respuesta del visualizador ante esta situación:

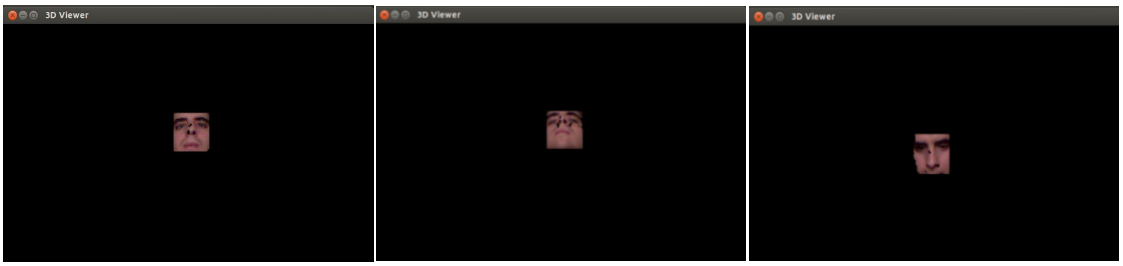
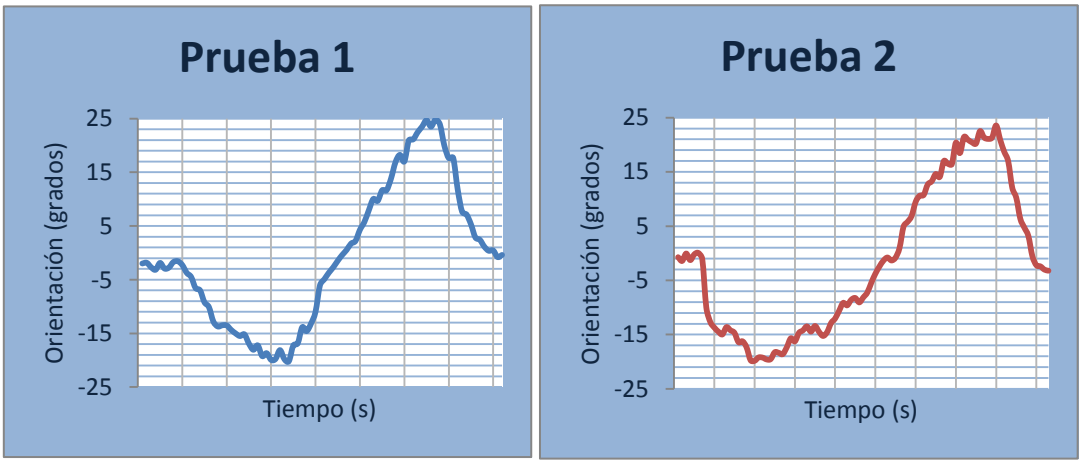


Figura 51. Visualizador movimiento Pitch.

Tal y como muestran las imágenes, la nube de puntos acompaña la cara durante su trayectoria y se queda únicamente con las facciones de esta para estudiar su comportamiento. Primeramente, cabe destacar cómo la cabeza comienza en una posición frontal para captar la referencia inicial, seguido de un desplazamiento de subida y otro de bajada. Dependiendo de la posición en la que se encuentre la cara se obtienen valores positivos (hacia abajo) o negativos (hacia arriba) del ángulo Pitch. Los datos extraídos en el fichero de texto se pueden comprobar en el *Anexo I* para cada una de las pruebas realizadas. Además, las siguientes gráficas (Véase Figura 52), ayudarán a comprender el funcionamiento de la plataforma durante la ejecución de este movimiento, con respecto del tiempo y para cada franja horaria determinada:



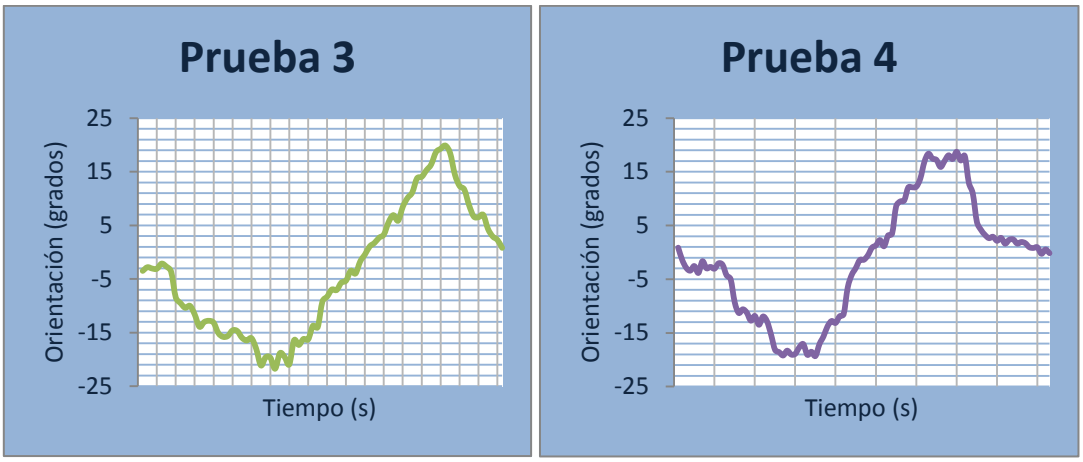


Figura 52. Gráficas movimiento Pitch (Véase *Anexo I*).

En estas representaciones, se puede comprobar cómo, en primer lugar, los valores experimentales comienzan a mantenerse en torno a los 0º (posición frontal). Posteriormente, se realiza un movimiento progresivo de la cabeza hacia arriba, donde se obtienen datos negativos hasta aproximadamente los -25º. Seguidamente, se observa un ascenso de las cifras debido a que se ha iniciado un descenso de la cabeza hasta llegar a resultados cercanos a los 25º, para finalizar volviendo al emplazamiento inicial. Tras analizar las 4 pruebas realizadas sacamos la conclusión de que, para estos parámetros iniciales, el sistema efectúa mediciones fiables en un rango de entre -25º y 25º.

5.3.2. Movimiento Yaw

Del mismo modo que el apartado anterior, se pone en marcha la aplicación y se comprobará su funcionamiento al describir un movimiento de rotación alrededor del eje Y. En las siguientes imágenes (Véase Figura 53), se puede observar la señal del visualizador para este caso:

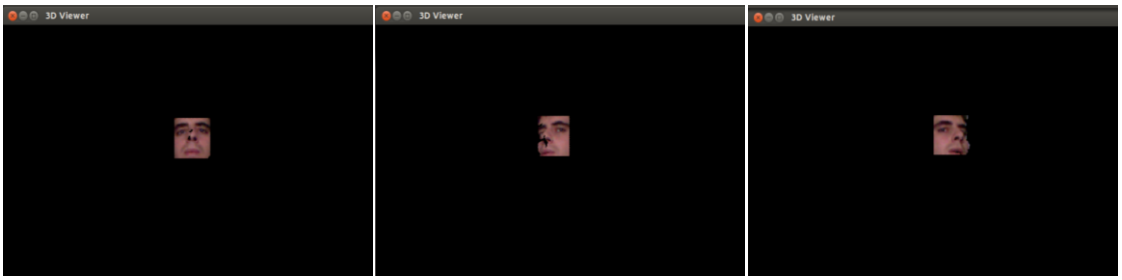


Figura 53. Visualizador movimiento Yaw.

Primeramente, hacer referencia a que se comenzará con la misma posición inicial que en el movimiento Pitch. El siguiente paso será describir una trayectoria lateral hacia la derecha el cual nos dará valores positivos y otro desplazamiento en sentido contrario para captar los datos negativos. En este caso, los resultados se podrán revisar en el *Anexo II*, a partir de los cuales se extrajeron las gráficas (Véase Figura 54) que aparecen a continuación:

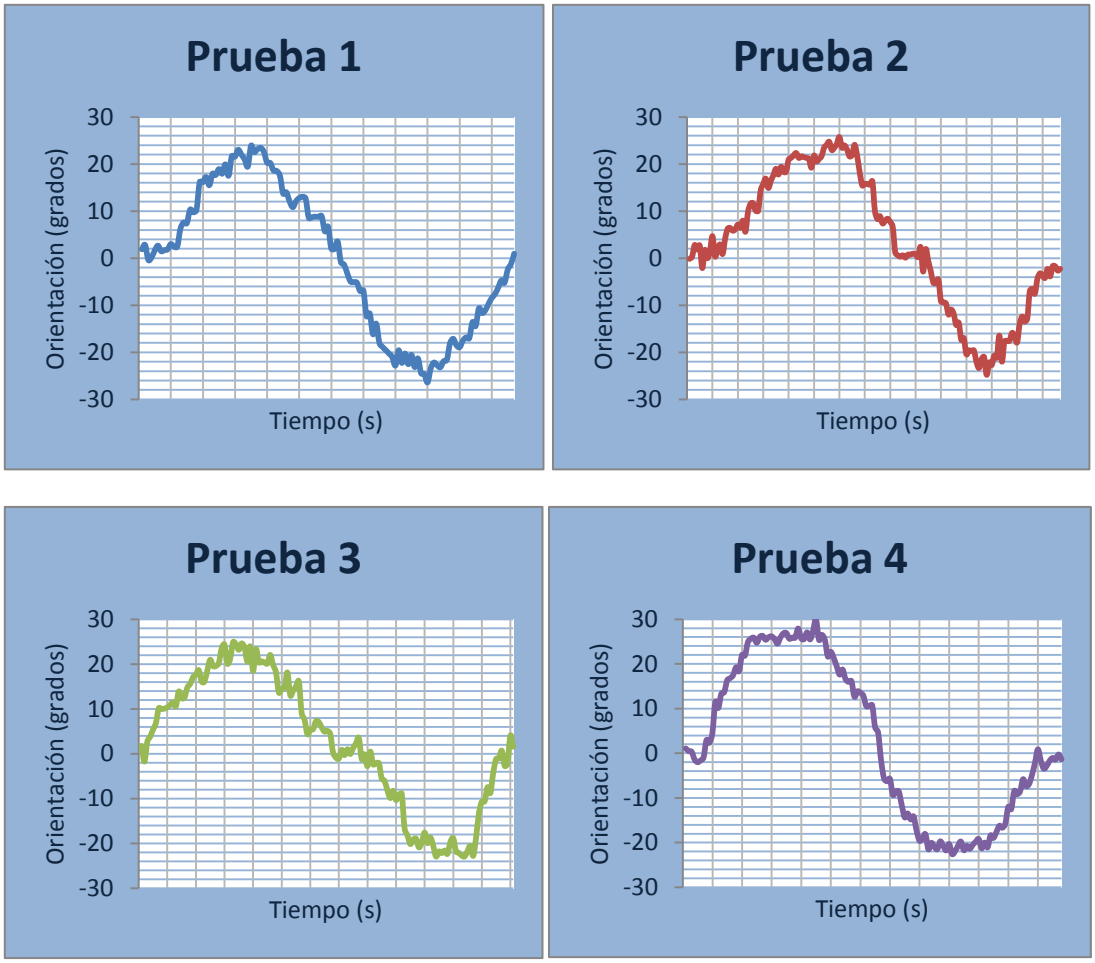


Figura 54. Gráficas movimiento Yaw (Véase *Anexo II*).

Como se puede apreciar, cada gráfico comienza con un pequeño periodo de tiempo donde se estabiliza la orientación, tomando valores cercanos a 0º y cogiendo como referencia la posición inicial. Acto seguido, se produce un desplazamiento lateral hacia la derecha del individuo recogiendo datos positivos hasta alcanzar una media aproximada superior a los 23º. De igual modo, se describirá un movimiento similar hacia la izquierda del conductor, obteniendo de esta forma un descenso de las cifras, alcanzando ángulos en torno los -23º. Finalmente, la cara volverá a su posición frontal. Tras analizar estos puntos, podemos concluir que, en base a las condiciones iniciales, el sistema tendrá un rango de entre -23º y 23º para el

movimiento Yaw. Fuera de estos límites, se han dado casos en los que el reconocimiento no era correcto puesto que percibía algún falso positivo al confundir la oreja con la nariz, lo que afectaría a las mediciones finales.

5.3.3. Movimiento Roll

Del mismo modo que en los casos anteriores, se pone en marcha la plataforma y, en este caso, estudiaremos el comportamiento del sistema al realizar un movimiento en torno al eje Z. En las siguientes capturas (Véase Figura 55) se muestran las señales extraídas del visualizador que nos ayudan a comprender su funcionamiento:

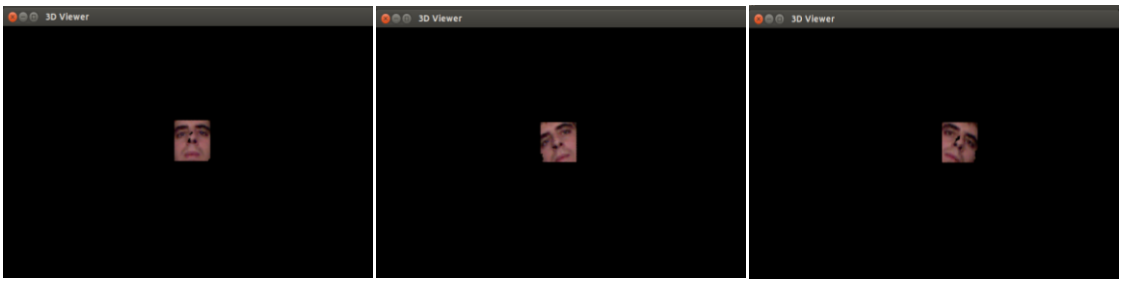


Figura 55. Visualizador movimiento Roll.

Este movimiento es el que más información aporta cuando se aprecia somnolencia en el conductor. Consiste en estudiar los desplazamientos de la cara al describir una trayectoria que comienza con una posición inicial (frontal), seguido de un movimiento oblicuo hacia la derecha (valores negativos) del automovilista y que finaliza con un desplazamiento similar en sentido contrario (valores positivos). Los datos captados, dependiendo de la franja horaria donde fueron sacados, aparecen en el *Anexo III*, los cuales han sido representados frente al tiempo en las siguientes gráficas (Véase Figura 56):

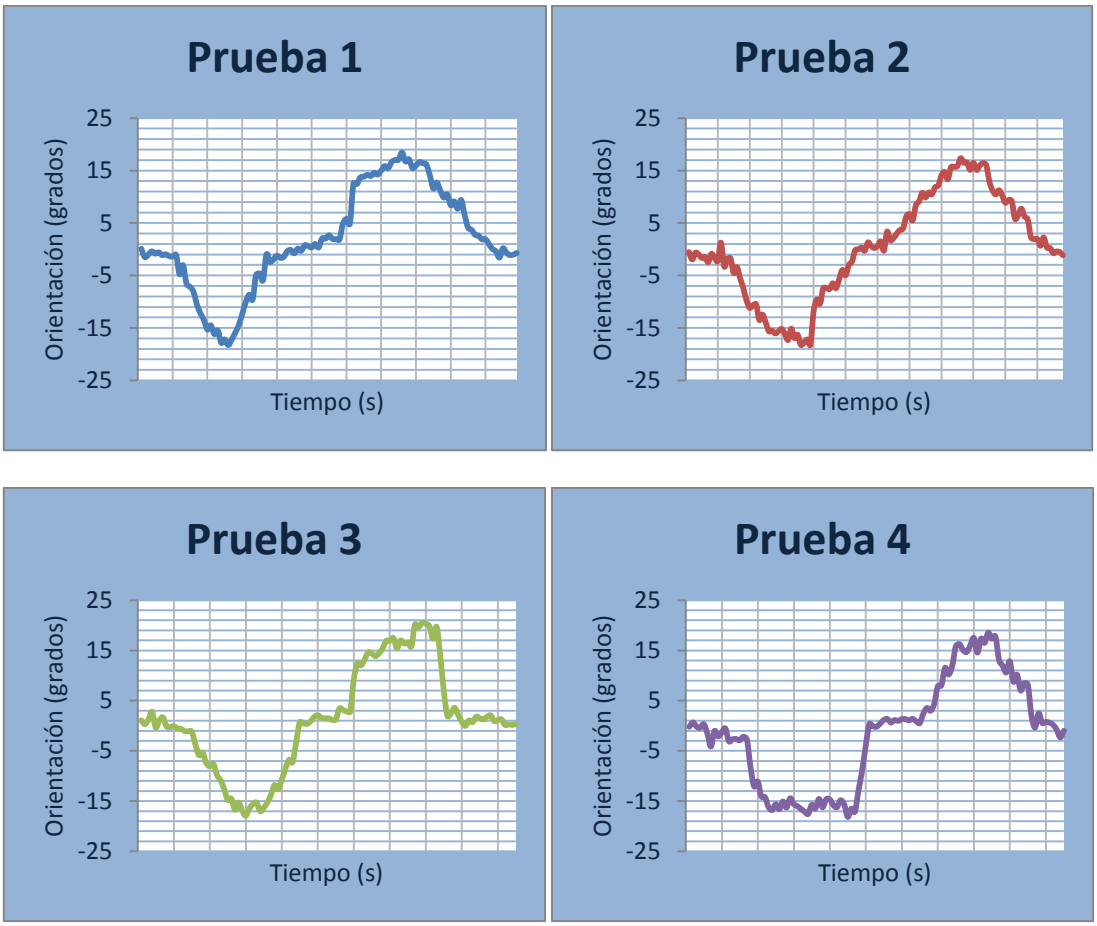


Figura 56. Gráficas movimiento Roll (Véase *Anexo III*).

Estas representaciones muestran los resultados prácticos obtenidos al realizar este movimiento. Como se puede observar, las gráficas comienzan y finalizan de la misma forma que en los casos anteriores, es decir, en la posición inicial donde se estabilizan los datos en torno a 0º. Una vez captada la referencia, se llevará a cabo un desplazamiento oblicuo hacia la derecha del conductor (a modo explicativo, este movimiento es similar al que se haría para mirar debajo de una mesa) registrando resultados negativos que alcanzarían los -20º. Posteriormente, se describirá una trayectoria idéntica en sentido contrario, percibiendo un ascenso en los valores que llegan a cifras aproximadas de 20º, para finalmente retornar al punto de inicio. A partir de estas mediciones y teniendo en cuenta los parámetros iniciales, se podrá concluir que la aplicación percibe datos fiables para el movimiento Roll en un rango aproximado de entre -20º y 20º.

Diseño de aplicación basada en Kinect para vehículo inteligente

Capítulo 6: Conclusiones

6.1. Conclusiones generales

Como se puede comprobar, ROS Hydro, junto con las librerías que incluye, nos permite crear aplicaciones de bajo y alto nivel que facilitan el trabajo del ingeniero a la hora de afrontar un determinado problema en el campo de la robótica y la visión artificial. Aunque se trata de un meta-sistema operativo complejo y de largo aprendizaje, es muy fácil de usar y la puesta en marcha de los diferentes ejecutables comunica de forma sencilla los topics, pudiendo trabajar con ellos en tiempo real.

Los objetivos iniciales del proyecto, se han llevado a cabo de forma satisfactoria, los resultados (gráficas y tablas) son coherentes, se encuentra completamente explicado con claridad y estructurado de tal forma que nos ayuda a comprender lo que ocurre en todo momento. Por este motivo, los 3 nodos están perfectamente diferenciados en función de su acometido. Lo que se ha querido es dar la posibilidad, a los investigadores de la Universidad, de poder manipularlo a partir del trabajo ya realizado.

Otro aspecto a recalcar, es que está creado para detectar una sola cara en el entorno, puesto que en la vida real se utilizará únicamente para analizar el comportamiento del conductor. En consecuencia, cuando se visualiza más de una cara en el entorno, solamente se extrae la información relativa a una de ellas, la que se ajuste de mejor forma a los parámetros marcados por el sistema.

Cabe destacar la labor de las bibliotecas Point Cloud Library y OpenCV para el tratamiento de las imágenes y nubes de puntos. Con ellas hemos podido manejar las señales en 3D, para obtener un mapa final que diera resultados válidos y fiables. También señalar la importancia del visualizador, el cual nos ha servido para comprender la realidad captada y analizar cómo se comportaba la plataforma. Realmente, este módulo no se encontrará posteriormente en el vehículo inteligente, pero servirá para observar y garantizar el correcto funcionamiento del proyecto.

Por último, explicar que este sensor tiene la ventaja de aportar datos de profundidad y color, aspecto que no se puede encontrar en las cámaras convencionales. Éstas últimas, solamente perciben el entorno en 2D, por lo tanto se perderían datos importantes de los objetos, mientras que con Kinect se puede extraer una representación tridimensional con suficiente información para hacer un estudio completo de las situaciones. Otra ventaja es que en cuanto a calidad/precio y prestaciones, este sensor puede contribuir a desarrollar sistemas más complejos con una inversión menor por lo que se reducirían gastos y se obtendrían resultados fiables. También, cabe destacar que este algoritmo podría adaptarse perfectamente para sistemas de visión estéreo o cámaras de tiempo de vuelo, que nos asegurarían un correcto funcionamiento con menos limitaciones de las que tiene Kinect (entre

las restricciones más importantes destacan mayor sensibilidad a la luz o que solo capta bien la señal en interiores, entre otras), pero encarecerían el proyecto y los resultados serían muy similares. No obstante, este trabajo ofrece mediciones precisas con un tiempo de procesamiento bastante bajo.

6.2. Principales dificultades y posibles mejoras

Este proyecto, se ha diseñado para abordar el problema propuesto basándose en unos parámetros iniciales definidos (posición/orientación del sensor constante e iluminación natural). Las posibles mejoras de esta aplicación se basarían en mejorar situaciones que no se correspondan con estas especificaciones.

Uno de los aspectos más importantes a tener en cuenta es que, antes de poner en marcha el sistema, se deberá mantener la cabeza en una posición frontal durante unos segundos. Cuando se observe que los datos se estabilizan, se podrán realizar los diferentes movimientos, pero siempre se tomará como referencia la posición inicial. En caso de no realizar este primer contacto de la cámara con el conductor, los resultados no se corresponderán con la realidad, es decir, no se estarán captando de forma óptima los datos.

Un gran avance, sería que funcionara perfectamente con exceso de iluminación (gran cantidad de reflejos) o en ambientes oscuros en el interior del vehículo (por ejemplo, de noche o en túneles):

- Exceso de iluminación: Este caso ocurre sobre todo cuando los rayos del sol inciden directamente al sensor desde un lateral o desde el fondo del vehículo. Esto afecta en gran medida a las diferentes señales que detecta la cámara y supondría un obstáculo a la hora de analizar el entorno (Véase Figura 57). [101]

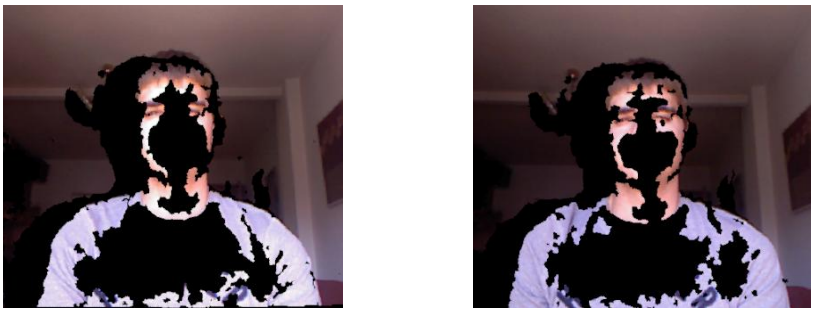


Figura 57. Funcionamiento con exceso de iluminación.

Este fenómeno también es debido a que Kinect capta la realidad mediante rayos infrarrojos, los cuales al distorsionarse o mezclarse con los rayos solares dificultan la correcta estimación de la posición de la cara. Este asunto es uno de los más difíciles de solucionar, puesto que es uno de los principales temas a investigar en la visión artificial actual.

- Visibilidad muy baja o nula en el interior del vehículo: El problema reside en que no capta correctamente la señal RGBA, por lo tanto, la correlación entre color y profundidad dificultaría la extracción de los datos. En casos con visibilidad baja, pero donde se aprecian el entorno y las facciones de la cara, el proyecto puede funcionar perfectamente, pero cuando se produce una visibilidad nula (Véase Figura 58), como puede ocurrir de noche en carreteras no iluminadas, este sistema no detecta ninguna cara.

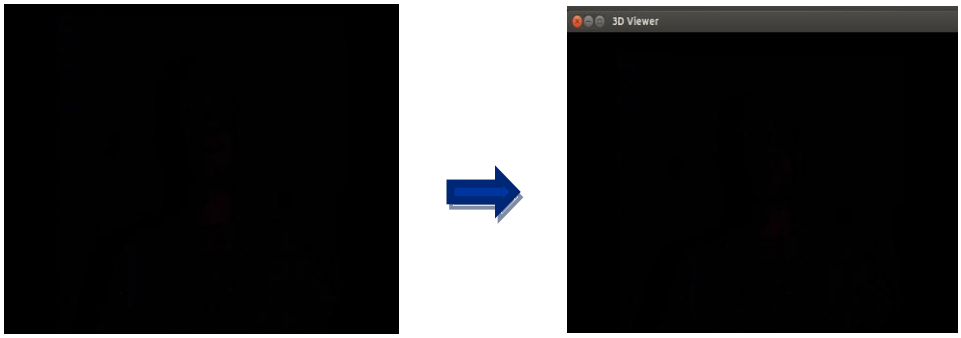


Figura 58. Funcionamiento con visibilidad nula.

Para resolver este fenómeno, se puede pensar en crear por ejemplo un *Modo nocturno*, en el que se utilicen únicamente los datos de profundidad. Para ello, se podría llevar a cabo la detección facial con otras técnicas de reconocimiento de objetos, primeramente realizando un estudio estadístico (mediante clasificador Bayesiano, funciones discriminantes, etc) y finalmente un agrupamiento de datos (usando Algoritmo de distancias encadenadas, Algoritmo K-medias, Algoritmo ISODATA, etc). [11]

Otra limitación de este sensor es que únicamente capta correctamente el entorno en interiores, puesto que necesita que los rayos infrarrojos choquen contra las paredes o cristales para detectar la profundidad. Por este motivo, una simple ventanilla bajada en el coche produciría una distorsión de la señal. Esto se podría solucionar utilizando algún sistema de visión estéreo o cámara de tiempo de vuelo, a las que les afecta menos este problema y son compatibles con el algoritmo, pero el proyecto se encarecería puesto que estos equipos suponen una mayor inversión.

También, comentar que, en ocasiones puntuales, se puede observar la presencia de falsos positivos (detectar la cara donde no se encuentra) o de falsos negativos, (no percibir la cara cuando está siendo visualizada). Los primeros son los más comunes de encontrar, con buena visibilidad, al superar ciertos ángulos en los que los errores son mayores (Véase Tabla 3). En cuanto a los segundos, se dan con menor asiduidad, pero se suelen producir en condiciones de escasez o exceso de iluminación. En la siguiente imagen (Véase Figura 59), se puede observar un falso positivo. En este ejemplo, la cara aparece en el entorno y no la localiza donde realmente tendría que hacerlo debido a que confunde la oreja con la nariz y, por lo tanto, realiza mediciones erróneas al llegar a un determinado ángulo. Para una mayor comprensión, se facilita una gráfica (Véase Figura 60), donde se aprecia perfectamente lo que ocurre. Los datos de esta representación se pueden encontrar en el *Anexo IV*.

Movimiento	Rango de funcionamiento*
Pitch	Entre -25 y 25 grados
Roll	Entre -20 y 20 grados
Yaw	Entre -23 y 23 grados

*En base a los parámetros iniciales y resultados obtenidos

Tabla 3. Rango de funcionamiento.

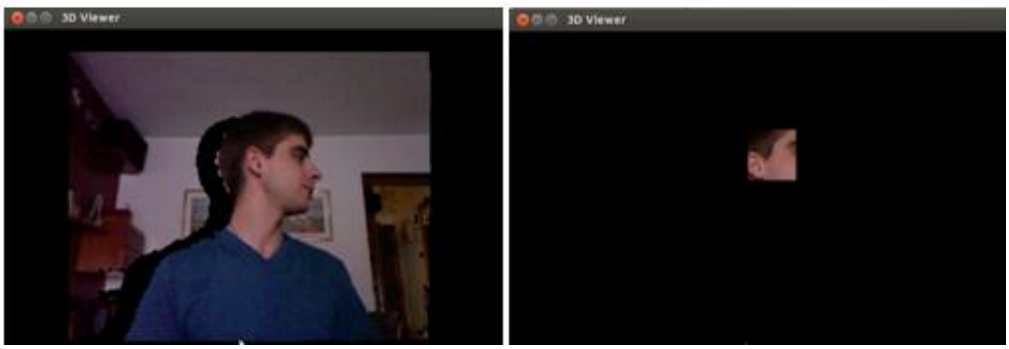


Figura 59. Falso positivo en movimiento Yaw.

Figura 60. Gráfica falso positivo en movimiento Yaw (Véase *Anexo IV*).

Otra posible novedad, puede ser monitorizar en el ordenador de a bordo la señal emitida por Kinect, para dar la posibilidad a cualquier usuario de ajustar la orientación del sensor en base a su estatura. La plataforma obtiene resultados fiables independientemente de la altura del conductor puesto que se ha probado a diferentes alturas, aunque solamente se hayan reflejado en este informe los datos extraídos en base a los parámetros iniciales.

La mejora que actualmente barajan los investigadores de la Universidad para este sistema es detectar el tiempo que permanecería el conductor con los ojos cerrados [3]. De esta forma, se podrá averiguar con mayor exactitud si el sujeto se encuentra dormido y durante cuánto tiempo para casos distracciones por somnolencia.

Para ajustar de mayor forma los datos a la vida real, se podrían acotar los ángulos con un máximo de 90º y un mínimo de -90º puesto que ninguna persona sería capaz de rotar más la cabeza. Por este motivo, todas mediciones que estén fuera de estos límites son considerados valores atípicos, es decir, que no son reales. Si se diera este problema, se podría solucionar obviando estos resultados y, en su lugar, repetir los valores anteriores donde se captaba bien la orientación porque es preferible apreciar un valor más parecido a la realidad que un dato imposible de aparecer (dentro de unos límites). Este tipo de situaciones no ocurren en la aplicación para ROS Hydro, pero de esta forma se evitarían en el caso de que ocurrieran.

Para finalizar, cabe destacar que el ajuste de los offset en los ejes vertical y horizontal, utilizan la herramienta *rqt_gui*. Esta interfaz, permite modificar los valores de cada desfase con el fin de que la nube de puntos quede perfectamente

alineada. Lo ideal, sería que esta función se hiciera de forma automática para no tener que modificar los datos cada vez que se inicia el sistema.

6.3. Futuras aplicaciones del proyecto

Este proyecto está enfocado a la investigación robótica, concretamente al campo de la visión artificial en la automoción. Las labores realizadas en él serán aprovechadas posteriormente por el Departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid para crear aplicaciones de mayor nivel.

Actualmente, este Departamento se encuentra desarrollando diferentes proyectos en un vehículo inteligente denominado IVVI 2.0 (Véase Figura 61). Todas estas plataformas, utilizan la visión artificial (mediante sensor Kinect, láser o cámaras estéreo, entre otras) para ayudar a mejorar la seguridad en los coches, evitar alcances, detección de peatones y señales, reconocimiento de carriles, etc.

Figura 61. Vehículo inteligente IVVI 2.0.

En este caso, la herramienta desarrollada tiene el principal objetivo de detectar la distracción del conductor al volante en ambientes con buena o baja visibilidad. Esto es muy útil a la hora de avisar de una falta de atención a la carretera.

Otra futura aplicación es la detección de la somnolencia a través de un cambio determinado en la orientación de la cara. Por este motivo, el sistema creado tendrá que detectar este cambio de la forma más rápida posible para permitir al conductor reaccionar con tiempo frente a un posible accidente. El único inconveniente, es que solamente funciona con buena o baja luminosidad, puesto que necesita captar correctamente los datos RGBA para su correcto desarrollo. En consecuencia, únicamente trabajará con datos fiables cuando no haya oscuridad, lo que supone un problema puesto que la mayoría de los accidentes por esta causa se producen de noche en carreteras no iluminadas. No obstante, este caso es ajeno al proyecto, pero es una de las principales dificultades que se encuentran actualmente en la visión artificial y que se continúan investigando.

Fuera del campo de la automoción, este sistema puede ayudar a desarrollar numerosas aplicaciones para discapacitados. Concretamente, puede ser práctico para acercar las nuevas tecnologías a personas parapléjicas, que solamente puedan mover la cabeza. Por ejemplo, podría servir para desarrollar alguna plataforma de mayor nivel que se adapte a televisiones inteligentes y que permitan modificar el canal o encender el dispositivo solamente realizando movimientos con la cara.

Como se ha podido apreciar, la creación de esta aplicación supone un pequeño avance en la visión artificial, pudiendo ser incorporado en nuevos sistemas para mejorar la calidad de vida y seguridad de las personas. No obstante, se continuara trabajando sobre ella para avanzar y seguir progresando en este campo.

Diseño de aplicación basada en Kinect para vehículo inteligente

Capítulo 7: Presupuesto

7.1. Información general

En este capítulo, se presenta el presupuesto necesario para realizar el proyecto. Se detallan tanto los costes de los diferentes equipos y materiales utilizados como los del personal. Añadir que la plataforma está enfocada al vehículo IVVI 2.0 (Intelligent Vehicle based on Visual Information), cuyo precio no aparece incluido en el presupuesto debido a que forma parte de varios proyectos diferentes.

7.2. Coste de equipos y material

En esta sección, introduciremos los costes de las distintas herramientas informáticas y equipos necesarios para desarrollar la aplicación.

En la siguiente tabla (Véase Tabla 4), se puede observar los costes del software utilizado en el transcurso del proyecto:

Código	Unidad de medida	Descripción	Medición	Coste unidad	Coste total
01		***SOFTWARE***			
01.01	ud.	ROS (Robot Operating System), meta-sistema operativo con licencia BSD. Software libre que incluye ROS en su versión Hydro para Ubuntu 12.04 LTS y otros paquetes y librerías necesarios compatibles con esta versión.	1	0,00 €	0,00 €
01.02	ud.	QtCreator 2.3, entorno de desarrollo de aplicaciones en distintos lenguajes de programación (C++, iOS, Android…). Herramienta gratuita en su versión para Ubuntu 12.04 LTS.	1	0,00 €	0,00 €
01.03	ud.	Microsoft Office 2007 Hogar y Estudiantes, instalado en Windows Vista. Está compuesto por los productos ofimáticos Microsoft Office Excel 2007, PowerPoint 2007, Word 2007 y OneNote 2007, para crear documentos, hojas de calculos y presentaciones.	1	120,00 €	120,00 €
			TOTAL SOFTWARE:		**120,00 €**

Tabla 4. Coste de software.

De la misma forma, la siguiente tabla (Véase Tabla 5) muestra el coste de los periféricos empleados para captar los datos y obtener los resultados finales:

Código	Unidad de medida	Descripción	Medición	Coste unidad	Coste total
02		***HARDWARE***			
02.01	ud.	Ordenador portátil AMILO Notebook Pi 3540 con una partición del disco duro en dos sistemas operativos: Windows Vista y Ubuntu 12.04 LTS.	1	550,00 €	550,00 €
02.02	ud.	Sensor Kinect desarrollado por Microsoft para XBOX 360. Periférico para adquisición de datos en 3D. Incluye transformador, conversor de 100/240 V de corriente alterna a 12 V de corriente continua.	1	83,00 €	83,00 €
			TOTAL HARDWARE:		**633,00 €**

Tabla 5. Coste de hardware.

7.3. Coste de personal

En este apartado, se aprecia el coste del personal (Véase Tabla 6) en base al tiempo de desarrollo de la plataforma:

Código	Unidad de medida	Descripción	Medición	Coste unidad	Coste total
03		***PERSONAL***			
03.01	mes	Graduado en Ingeniería Electrónica Industrial y Automática	4	1.500 €/mes	6.000,00 €
			TOTAL PERSONAL:		**6.000,00 €**

Tabla 6. Coste de personal.

7.4. Coste total del proyecto

El desarrollo de esta aplicación ha supuesto un coste total de:

Código	Concepto	Importe
01	Coste software	120,00 €
02	Coste hardware	633,00 €
03	Coste personal	6.000,00 €
TOTAL DEL PROYECTO:		**6.753,00 €**

Tabla 7. Coste total del proyecto.

Diseño de aplicación basada en Kinect para vehículo inteligente

Capítulo 8: Bibliografía

Bibliografía

[1] «ROS Hydro,» [En línea]. Available: http://wiki.ros.org/hydro. [Último acceso: 10 Septiembre 2014].

[2] «Sensor Kinect,» [En línea]. Available: http://www.xbox.com/es-ES/Kinect. [Último acceso: 21 Julio 2014].

[3] G. Peláez, F. García, J. Armingol y A. d. l. Escalera, «Driver Monitoring Based on Low-Cost 3D Sensor,» *IEEE Transactions on Intelligent Transportation Systems,* vol. 15, nº 4, pp. 1855-1860, 2014.

[4] «Ubuntu,» [En línea]. Available: http://www.ubuntu.com/. [Último acceso: 20 Octubre 2013].

[5] «OpenCV,» [En línea]. Available: http://opencv.org/. [Último acceso: 10 Septiembre 2014].

[6] «Point Cloud Library,» [En línea]. Available: http://pointclouds.org/. [Último acceso: 10 Septiembre 2014].

[7] «Real Academia Española,» [En línea]. Available: http://www.rae.es/. [Último acceso: 3 Septiembre 2014].

[8] S. M. Kosslyn, Image and Brain, A. Bradford Book, 1994.

[9] D. N. C. L. Grabner, «La importancia del sentido de la vista en la educación y el aprendizaje,» *Asociación Educar. Ciencias y Neurociencias aplicadas al Desarrollo Humano,* 2012.

[10] J. González, Visión por computador, Paraninfo, 1999.

[11] A. d. l. Escalera, VISIÓN POR COMPUTADOR. Fundamentos y métodos, Madrid: Prentice Hall, 2001.

[12] «Multi-sensing system with rear camera,» Nissan Motor Corporation, [En línea]. Available: http://www.nissan-global.com/EN/TECHNOLOGY/OVERVIEW/rear_camera.html. [Último acceso: 11 Septiembre 2014].

[13] «Soluciones de Visión Artificial para Automoción,» Infaimon Visión Artificial, [En línea]. Available: http://blog.infaimon.com/2011/12/soluciones-de-vision-artificial-para-automocion/. [Último acceso: 11 Septiembre 2014].

[14] D. H. Ballard y C. M. Brown, Computer Vision, Prendice-Hall, 1982.

[15] S. Horowitz y T. Pavlidis, «Picture segmentation by a directed split-and-merge procedure,» de *The 2nd International Joint Conference on Pattern Recognition*, Copenhagen, 1974.

[16] J. Serra, «Image Analysis and Mathematical Morphology,» *Academic Press,* vol. I, 1982.

[17] F. Meyer y S. Beucher, «Morphological segmentation,» *J. of Visual Communication & Image Representation,* vol. 1, pp. 21-45, 1990.

[18] A. Jain, Fundamentals of Digital Image Processing, New Jersey: Prentice Hall, 1989.

[19] C. Kapoutsis, C. Vavoulidis y I. Pitas, «Morphological iterative closest point algorithm,» *IEEE Transactions on Image Processing,* vol. 8, nº 11, pp. 1644 - 1646, 1999.

[20] «Downsampling a PointCloud using a VoxelGrid filter,» Point Cloud Library, [En línea]. Available: http://www.pointclouds.org/documentation/tutorials/voxel_grid.php. [Último acceso: 20 Agosto 2014].

[21] «Image and Signal Processing,» *Lecture Notes in Computer Science,* vol. 6134, pp. 200-209, 2010.

[22] E. Hjelmas y J. Wroldsen, «Recognizing Faces from the Eyes Only,» de *The 11th Scandinavian Conference on Image Analysis*, 1999.

[23] K. I. Chang, K. W. Bowyer, P. J. Flynn y S. Member, «Multiple nose region matching for 3D face recognition under varying facial expression,» de *IEEE Transaction on Pattern Analysis and Machine Intelligence 28*, 2006.

[24] P. Viola y M. Jones, «Rapid object detection using a boosted cascade of simple features,» de *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[25] C. Zhang y Z. Zhang, «A Survey of Recent Advances in Face detection,» 2010.

[26] «3D TUTORIALS,» [En línea]. Available: http://all3dtutorials.blogspot.com.es/2010/08/there-are-three-types-of-3d-modeling.html. [Último acceso: 15 Septiembre 2014].

[27] «3D Modeling Basics,» PeachPit, [En línea]. Available: http://www.peachpit.com/articles/article.aspx?p=30594. [Último acceso: 15 Septiembre 2014].

[28] L. S. Wolfe y P.E., «9 criterios para elegir un sistema de CAD en 3D,» [En línea]. Available: http://all3dtutorials.blogspot.com.es/2010/08/there-are-three-types-of-3d-modeling.html. [Último acceso: 15 Septiembre 2014].

[29] Y. Cui, S. Schuon, D. Chan, S. Thrun y C. Theobalt, «3D shape scanning with a time-of-flight camera,» de *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010 .

[30] «Cámara estereoscópica,» Ingeniatic, [En línea]. Available: http://ingeniatic.euitt.upm.es/index.php/tecnologias/item/394-c%C3%A1mara-estereosc%C3%B3pica. [Último acceso: 15 Septiembre 2014].

[31] L. Gerhardt y K. Hyun, «View planning applied to coordinate measuring machine (CMM) measurement,» de *International IEEE/IAS Conference on Industrial Automation and Control: Emerging Technologies*, 1995.

[32] «Software ROS,» Willow Garage, [En línea]. Available: https://www.willowgarage.com/pages/software/ros-platform. [Último acceso: 12 Septiembre 2014].

[33] «ROS (Robot Operating System),» [En línea]. Available: http://www.ros.org/. [Último acceso: 12 Septiembre 2014].

[34] «ROS Tutorials,» [En línea]. Available: http://wiki.ros.org/ROS/Tutorials. [Último acceso: 12 Septiembre 2014].

[35] «ROS Answers,» [En línea]. Available: http://answers.ros.org/questions/. [Último acceso: 12 Septiembre 2014].

[36] «ROS Indigo Igloo,» [En línea]. Available: http://wiki.ros.org/indigo. [Último acceso: 12 Septiembre 2014].

[37] «ROS Groovy,» [En línea]. Available: http://wiki.ros.org/groovy. [Último acceso: 12 Septiembre 2014].

[38] «ROS Fuerte,» [En línea]. Available: http://wiki.ros.org/fuerte. [Último acceso: 12 Septiembre 2014].

[39] «ROS Electric,» [En línea]. Available: http://wiki.ros.org/electric. [Último acceso: 12 Septiembre 2014].

[40] «ROS Diamondback,» [En línea]. Available: http://wiki.ros.org/diamondback. [Último acceso: 12 Septiembre 2014].

[41] «ROS C Turtle,» [En línea]. Available: http://wiki.ros.org/cturtle. [Último acceso: 12 Septiembre 2014].

[42] «ROS Box Turtle,» [En línea]. Available: http://wiki.ros.org/boxturtle. [Último acceso: 12 Septiembre 2014].

[43] «Catkin,» [En línea]. Available: http://wiki.ros.org/catkin. [Último acceso: 12 Septiembre 2014].

[44] «Target Platforms,» [En línea]. Available: http://www.ros.org/reps/rep-0003.html. [Último acceso: 12 Septiembre 2014].

[45] «Rosbuild,» [En línea]. Available: http://wiki.ros.org/rosbuild. [Último acceso: 12 Septiembre 2014].

[46] J. M. O'Kane, A Gentle Introduction to ROS, Paperback, 2013.

[47] «ROS Concepts,» [En línea]. Available: http://wiki.ros.org/ROS/Concepts. [Último acceso: 12 Septiembre 2014].

[48] «ROS Packages,» [En línea]. Available: http://wiki.ros.org/Packages. [Último acceso: 12 Septiembre 2014].

[49] «ROS Metapackages,» [En línea]. Available: http://wiki.ros.org/Metapackages. [Último acceso: 12 Septiembre 2014].

[50] «ROS Package Manifests,» [En línea]. Available: http://wiki.ros.org/catkin/package.xml. [Último acceso: 12 Septiembre 2014].

[51] «ROS Repositories,» [En línea]. Available: http://wiki.ros.org/RecommendedRepositoryUsage/CommonGitHubOrganizations. [Último acceso: 12 Septiembre 2014].

[52] «ROS msg,» [En línea]. Available: http://wiki.ros.org/msg. [Último acceso: 12 Septiembre 2014].

[53] «ROS srv,» [En línea]. Available: http://wiki.ros.org/srv. [Último acceso: 12 Septiembre 2014].

[54] «ROS Nodes,» [En línea]. Available: http://wiki.ros.org/Nodes. [Último acceso: 12 Septiembre 2014].

[55] «ROS Master,» [En línea]. Available: http://wiki.ros.org/Master. [Último acceso: 12 Septiembre 2014].

[56] «ROS Parameter Server,» [En línea]. Available: http://wiki.ros.org/Parameter%20Server. [Último acceso: 12 Septiembre 2014].

[57] «ROS Messages,» [En línea]. Available: http://wiki.ros.org/Messages. [Último acceso: 12 Septiembre 2014].

[58] «ROS Topics,» [En línea]. Available: http://wiki.ros.org/Topics. [Último acceso: 12 Septiembre 2014].

[59] «ROS Services,» [En línea]. Available: http://wiki.ros.org/Services. [Último acceso: 12 Septiembre 2014].

[60] «ROS Bags,» [En línea]. Available: http://wiki.ros.org/Bags. [Último acceso: 12 Septiembre 2014].

[61] «ROS Distributions,» [En línea]. Available: http://wiki.ros.org/Distributions. [Último acceso: 12 Septiembre 2014].

[62] «ROS Stacks,» [En línea]. Available: http://wiki.ros.org/action/show/rosbuild/Stacks?action=show&redirect=Stacks. [Último acceso: 12 Septiembre 2014].

[63] «ROS Wiki,» [En línea]. Available: http://wiki.ros.org/es. [Último acceso: 12 Septiembre 2014].

[64] «ROS Tickets,» [En línea]. Available: http://wiki.ros.org/Tickets. [Último acceso: 12 Septiembre 2014].

[65] «ROS Mailing List,» [En línea]. Available: http://wiki.ros.org/action/show/Support?action=show&redirect=Mailing+Lists. [Último acceso: 12 Septiembre 2014].

[66] «ROS Blog,» [En línea]. Available: http://www.ros.org/news/. [Último acceso: 12 Septiembre 2014].

[67] «Catkin or Rosbuild,» [En línea]. Available: http://wiki.ros.org/catkin_or_rosbuild. [Último acceso: 12 Septiembre 2014].

[68] «Rosbuild CMakeList.txt,» [En línea]. Available: http://wiki.ros.org/rosbuild/CMakeLists.txt. [Último acceso: 12 Septiembre 2014].

[69] «Catkin CMakeList.txt,» [En línea]. Available: http://wiki.ros.org/catkin/CMakeLists.txt. [Último acceso: 12 Septiembre 2014].

[70] R. Rusu y S. Cousins, «3D is here: Point Cloud Library (PCL),» de *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[71] «ROS Nodelet,» [En línea]. Available: http://wiki.ros.org/nodelet. [Último acceso: 12 Septiembre 2014].

[72] «Software OpenCV,» [En línea]. Available: https://www.willowgarage.com/pages/software/opencv. [Último acceso: 13 Septiembre 2014].

[73] G. Bradski y A. Kaehler, Learning OpenCV. Computer Vision with the OpenCV Library, O'Reilly Media, 2008.

[74] «ROS-OpenCV CvBridge,» [En línea]. Available: http://wiki.ros.org/cv_bridge. [Último acceso: 13 Septiembre 2014].

[75] «OpenCV CvMat,» [En línea]. Available: http://physics.nyu.edu/grierlab/manuals/opencv/classcv_1_1Mat.html. [Último acceso: 13 Septiembre 2014].

[76] «Lenguaje de programación C++,» [En línea]. Available: http://www.cplusplus.com/. [Último acceso: 13 Septiembre 2014].

[77] «QtCreator,» [En línea]. Available: http://qt-project.org/wiki/category:Tools::QtCreator. [Último acceso: 13 Septiembre 2014].

[78] «ROS freenect_stack,» [En línea]. Available: http://wiki.ros.org/freenect_stack. [Último acceso: 14 Septiembre 2014].

[79] M. Tölgyessy y P. Hubinský, «The Kinect Sensor in Robotics Education,» *Innoc.*

[80] «Kinect calibration,» [En línea]. Available: http://wiki.ros.org/kinect_calibration/technical. [Último acceso: 14 Septiembre 2014].

[81] «Intrinsic Calibration,» [En línea]. Available: http://wiki.ros.org/openni_launch/Tutorials/IntrinsicCalibration. [Último acceso: 14 Septiembre 2014].

[82] «camera_calibration,» [En línea]. Available: http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration. [Último acceso: 14 Septiembre 2014].

[83] «ROS rqt_reconfigure,» [En línea]. Available: http://wiki.ros.org/rqt_reconfigure. [Último acceso: 14 Septiembre 2014].

[84] «RACE y BP-CASTROL contra las distracciones al volante,» 29 Mayo 2013. [En línea]. Available: http://www.bp.com/es_es/spain/prensa/notas-de-prensa/2013/race-y-bp-castrol-contra-las-distracciones-al-volante.html. [Último acceso: 1 Septiembre 2014].

[85] J. Serrano, «Las distracciones más frecuentes al volante,» 31 Marzo 2014. [En línea]. Available: http://www.autopista.es/reportajes/articulo/distracciones-frecuentes-volante-accidentes-96524. [Último acceso: 1 Septiembre 2014].

[86] «pcl::PointXYZRGBA Struct Reference,» [En línea]. Available: http://docs.pointclouds.org/1.7.0/structpcl_1_1_point_x_y_z_r_g_b_a.html. [Último acceso: 2 Septiembre 2014].

[87] «ROS sensor_msgs/PointCloud2,» [En línea]. Available: http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html. [Último acceso: 3 Septiembre 2014].

[88] «ROS sensor_msgs/Image,» [En línea]. Available: http://docs.ros.org/api/sensor_msgs/html/msg/Image.html. [Último acceso: 3 Septiembre 2014].

[89] «Using CvBridge To Convert Between ROS Images And OpenCV Images,» [En línea]. Available: http://wiki.ros.org/cv_bridge/Tutorials/UsingCvBridgeToConvertBetweenROSImagesAndOpenCVImages. [Último acceso: 2 Septiembre 2014].

[90] «Distancia focal,» The Web Photo, [En línea]. Available: http://www.thewebfoto.com/2-hacer-fotos/203-distancia-focal. [Último acceso: 2 Septiembre 2014].

[91] «ROS rqt_gui,» [En línea]. Available: http://wiki.ros.org/rqt_gui. [Último acceso: 2 Septiembre 2014].

[92] «Ángulos de Euler,» Wikipedia, [En línea]. Available: http://es.wikipedia.org/wiki/%C3%81ngulos_de_Euler.

[93] «Mínimos Cuadrados,» Wikipedia, [En línea]. Available: http://es.wikipedia.org/wiki/M%C3%ADnimos_cuadrados. [Último acceso: 3 Septiembre 2014].

[94] «PCL Visualizer,» [En línea]. Available: http://pointclouds.org/documentation/tutorials/pcl_visualizer.php. [Último acceso: 3 Septiembre 2014].

[95] «Point Cloud Library Visualization,» [En línea]. Available: http://docs.pointclouds.org/trunk/group__visualization.html. [Último acceso: 3 Septiembre 2014].

[96] «ROS rqt_graph,» [En línea]. Available: http://wiki.ros.org/rqt_graph. [Último acceso: 3 Septiembre 2014].

[97] «ROS geometry_msgs/Point,» [En línea]. Available: http://docs.ros.org/api/geometry_msgs/html/msg/Point.html. [Último acceso: 3 Septiembre 2014].

[98] «Movimiento Pitch. Diseño de aplicación basada en Kinect para vehículo inteligente,» Youtube, [En línea]. Available: https://www.youtube.com/watch?v=6LbVcJLnzW8. [Último acceso: 22 Septiembre 2014].

[99] «Movimiento Yaw. Diseño de aplicación basada en Kinect para vehículo inteligente,» Youtube, [En línea]. Available: https://www.youtube.com/watch?v=ZQ4DSfD0KZA. [Último acceso: 22 Septiembre 2014].

[100] «Movimiento Roll. Diseño de aplicación basada en Kinect para vehículo inteligente,» Youtube, [En línea]. Available: https://www.youtube.com/watch?v=SENFxfcd6b0. [Último acceso: 22 Septiembre 2014].

[101] «Solucionar problemas de seguimiento del cuerpo,» XBOX, [En línea]. Available: http://support.xbox.com/es-ES/xbox-360/kinect/body-tracking-troubleshoot. [Último acceso: 4 Septiembre 2014].

Diseño de aplicación basada en Kinect para vehículo inteligente

Anexo I. Datos movimiento Pitch

Prueba 1

	Pitch	Roll	Yaw		Pitch	Roll	Yaw
1	-1.998396	0.363851	2.552069	42	-4.917102	-2.778431	4.656434
2	-1.856789	0.131504	0.59998	43	-3.714711	0.642113	1.119606
3	-2.700941	0.596825	2.80267	44	-2.704956	-0.403312	4.32403
4	-3.166265	0.337975	1.948115	45	-1.493558	0.555541	-2.621009
5	-1.880991	0.493899	3.792771	46	-0.41583	0.312389	-0.280815
6	-2.93895	-0.593948	2.267135	47	0.52783	-2.000965	7.570696
7	-2.752841	-0.893944	0.30997	48	1.703215	-2.648417	7.009539
8	-1.727035	-0.943015	1.797527	49	2.197445	-1.430408	3.26596
9	-1.569007	-0.145074	1.533555	50	4.27612	-3.921112	5.294432
10	-2.282707	-0.059231	0.053489	51	5.731379	-2.468146	6.529318
11	-3.725174	-0.085282	2.947776	52	7.911711	-2.683137	6.736401
12	-4.50381	0.544666	2.779375	53	10.01939	-2.548843	7.466734
13	-6.580956	0.128169	0.492157	54	9.666264	-2.510601	6.887065
14	-6.984018	0.660161	-1.083166	55	11.63694	-1.818522	5.466681
15	-9.215964	-0.613558	-1.864566	56	11.64807	-2.076753	6.210222
16	-10.10735	0.871643	-1.986174	57	13.82424	-2.754021	5.876328
17	-12.80068	0.145356	-2.064491	58	16.86138	-1.614742	8.959514
18	-13.7277	-2.820501	1.870205	59	18.24117	2.570885	1.107483
19	-13.51188	-0.206666	-1.635666	60	17.01734	-2.584235	6.725048
20	-13.5242	0.536935	-2.845533	61	20.92421	2.269344	0.93474
21	-14.35496	2.034275	-3.74468	62	21.11341	0.527291	0.955915
22	-15.01716	0.873023	-2.447891	63	22.45399	-3.900527	4.610442
23	-15.51249	2.382395	-1.545048	64	23.49968	-2.370484	4.186064
24	-15.19438	2.11368	-2.083431	65	24.70027	-2.257183	3.357304
25	-16.90187	1.751296	-3.036919	66	23.47765	-1.991386	4.241306
26	-18.07166	1.103192	-3.380249	67	24.80078	-0.85256	3.149228
27	-17.24221	3.628126	-7.832444	68	23.93056	2.251429	-1.409177
28	-19.25337	1.982436	-2.319472	69	19.97806	-2.692913	6.342381
29	-18.69687	2.553052	-2.284449	70	17.62093	-1.682395	5.428451
30	-19.94974	2.821852	-2.834603	71	17.67948	-2.124514	5.467162
31	-19.77549	2.193238	-3.384865	72	11.83625	-0.955475	5.087341
32	-18.12148	3.121652	-8.685455	73	7.655243	-2.91418	6.79597
33	-19.83286	2.386324	-4.035804	74	7.129172	-2.696113	6.139265
34	-20.19989	3.852589	-3.645808	75	5.30637	-3.19743	3.810166
35	-17.25269	3.320799	-3.369091	76	2.84456	-1.806576	4.634353
36	-16.87772	2.92706	-3.83409	77	2.425272	-1.069004	3.300497
37	-13.87036	2.95224	-3.83797	78	1.148932	0.848569	-3.751548
38	-14.59569	-0.495944	-2.085415	79	0.37126	-2.035745	3.954287
39	-13.16912	-0.842918	-1.494272	80	0.442505	-0.640105	3.048472
40	-11.01664	-2.415424	4.317324	81	-0.804191	-0.950899	5.077702
41	-6.06329	0.223753	-1.975127	82	-0.401294	-1.595013	5.009575

Prueba 2

	Pitch	Roll	Yaw
1	-0.726904	-1.853206	4.109188
2	-1.421786	-2.05578	4.247223
3	-0.046124	-0.657241	2.988787
4	-1.274096	-1.684439	3.352525
5	-0.221082	-1.186023	2.741693
6	0.067492	-1.733519	4.400682
7	-1.06139	-0.706707	3.569502
8	-9.978373	-0.338323	1.289183
9	-12.68824	-2.374377	3.238157
10	-13.71439	-1.502492	2.019317
11	-14.4635	-0.643261	1.147392
12	-14.97596	-0.235888	1.535983
13	-13.63054	0.953365	-2.219194
14	-14.21443	-0.443568	0.329301
15	-14.68271	0.113979	0.998747
16	-16.40346	0.889412	-2.53855
17	-16.24055	1.530411	0.967089
18	-17.35	2.375428	-4.961366
19	-19.73566	1.346282	0.369383
20	-19.83864	0.89397	1.605518
21	-19.21299	0.754761	-0.57827
22	-19.25297	0.832739	-1.228626
23	-19.58856	1.097358	-1.576744
24	-19.52734	1.023624	-10.19002
25	-18.25957	0.60087	0.198769
26	-18.37069	5.955893	-4.079306
27	-18.61002	2.238292	0.754238
28	-17.33339	0.806726	2.421954
29	-15.67988	1.241335	-10.0987
30	-16.25722	0.882119	-10.16246
31	-14.69753	1.041961	-10.41238
32	-14.23684	4.665199	-2.981108
33	-13.48856	1.916589	-5.775143
34	-14.46875	5.863021	-1.751368
35	-13.40729	4.584415	-8.654414
36	-14.4872	4.201641	-2.352689
37	-15.25339	0.324803	-0.325439
38	-14.55946	-1.260805	1.519595
39	-12.86558	0.033546	0.547094
40	-12.04097	0.243806	1.072825
41	-10.60013	-1.363475	-0.139413
42	-9.156109	-0.689722	0.012682
43	-9.609393	8.843983	-9.666339
44	-8.58878	7.723269	-9.131894
45	-8.250359	6.482349	-9.849849
46	-9.072444	6.051246	-9.940133

	Pitch	Roll	Yaw
47	-8.147213	-0.719423	-0.643672
48	-7.239774	-0.057499	1.833333
49	-5.420118	0.519956	6.092077
50	-3.769258	-0.829814	2.957463
51	-2.459696	-1.220674	3.823618
52	-1.360079	-0.767412	1.200715
53	-0.785582	-1.372923	2.577921
54	-1.336495	-1.042955	3.392693
55	-0.811712	-0.712004	3.224357
56	0.966843	-0.673864	6.134421
57	4.916836	-1.180071	5.386356
58	5.788331	-2.657274	7.523766
59	6.881166	-2.163476	7.400248
60	9.466569	-1.05755	5.908287
61	10.61036	-1.025944	6.672167
62	10.74527	-0.05057	3.103678
63	12.67843	-0.047576	4.021662
64	13.27231	-2.75543	6.261077
65	14.61684	-0.304314	4.176764
66	14.0783	-1.772414	2.393957
67	16.99363	0.365705	4.33942
68	16.51373	-1.365999	3.505553
69	16.43322	-1.365776	1.099017
70	20.39731	-0.071456	2.506178
71	18.45943	-0.792442	2.319486
72	21.42946	-0.874322	0.371325
73	20.92744	0.080589	3.265139
74	20.45405	-0.18178	3.125424
75	20.23585	-2.070684	2.014749
76	22.53306	3.064143	-2.498554
77	21.34921	-0.281258	0.317096
78	21.09338	-1.224282	1.40969
79	21.33338	-1.732371	2.208108
80	23.5713	1.01019	4.861562
81	20.9511	-0.471824	4.745838
82	18.72962	-0.102632	3.407976
83	16.95986	-0.537141	7.022795
84	12.08883	-0.437266	4.411735
85	10.23849	-1.492485	4.011765
86	6.362874	-1.943804	3.921399
87	4.731929	-1.912405	6.380501
88	3.19912	-1.34976	2.616545
89	-0.536714	-2.684292	3.528133
90	-2.217415	0.182901	0.648189
91	-2.416579	-0.147305	2.399472
92	-3.024745	0.558651	1.343841

	Pitch	Roll	Yaw
93	-3.224974	0.236572	1.250878

Prueba 3

	Pitch	Roll	Yaw
1	-3.518657	0.33966	1.029477
2	-2.827786	0.596578	0.661232
3	-3.071497	1.256674	1.097873
4	-3.119319	1.119113	1.34062
5	-2.150362	0.17151	1.375901
6	-2.696277	0.089037	1.761313
7	-3.617701	-0.078195	0.390915
8	-8.474215	-1.299338	1.127021
9	-9.529093	-2.96444	-0.129547
10	-10.41233	-0.639284	-1.080187
11	-10.02847	0.839706	-0.29286
12	-11.63277	-0.337454	1.8768
13	-13.94368	-0.659417	0.401736
14	-13.05745	0.424082	0.732036
15	-12.82259	-0.084047	-1.864055
16	-13.17753	8.377461	-2.711532
17	-15.11763	-0.1324	0.417986
18	-15.8217	0.592081	1.440461
19	-15.66211	1.125128	-0.73947
20	-14.61241	7.128649	-2.489142
21	-14.73424	8.700716	-3.237538
22	-15.99213	3.130781	-8.685076
23	-16.50313	7.460576	-2.440706
24	-16.06442	0.90633	-1.676811
25	-18.04696	3.105357	-2.669117
26	-21.15959	0.936715	1.082273
27	-19.6186	1.077138	2.583349
28	-19.70831	0.607502	1.890239
29	-21.77437	3.315694	1.259327
30	-18.87978	0.403325	2.685936
31	-19.54514	0.329017	2.202943
32	-20.95188	3.740404	-3.303475
33	-16.45357	-3.629431	3.982153
34	-17.34067	0.819958	3.395113
35	-16.20494	-0.922236	2.80605
36	-16.28556	0.858198	2.66371
37	-13.74693	-0.096313	3.346699
38	-14.02653	-1.338354	4.987329
39	-9.219995	-1.149257	5.173521
40	-8.319998	-1.33289	4.972343
41	-6.976003	-1.761748	3.578271
42	-7.095213	-2.379966	4.796113
43	-5.705004	-1.232602	5.674671
44	-5.331943	-1.18111	4.596077
45	-3.474167	-0.257907	6.236723
46	-4.010781	-1.326473	2.03022
47	-1.725313	-0.482701	6.845177
48	-0.440793	-1.537896	4.725261
49	1.045813	-1.262475	5.30945
50	1.699892	-1.776995	5.507081
51	2.687098	-1.421048	6.418207
52	3.270552	-0.755785	6.12413
53	5.586192	-2.223371	6.885368
54	6.899992	-1.698289	7.996951
55	5.843117	-1.356257	2.886576
56	8.45078	-0.132308	9.160412
57	10.10038	-1.469142	4.99865
58	11.12303	-1.205943	3.522248
59	13.75776	-0.436214	5.789711
60	14.03748	-2.545809	6.581325
61	15.2689	-1.577076	5.728778
62	16.3439	-1.67288	4.127859
63	18.62805	0.062181	6.75102
64	19.27028	-1.547021	5.491118
65	19.8779	-1.761523	5.283605
66	18.46272	-1.066707	5.357587
67	14.47125	-0.786291	5.542166
68	12.33119	-0.842665	4.660033
69	11.67343	-1.141838	5.329389
70	8.759141	-1.298023	4.689484
71	6.6276	-2.118757	5.313875
72	6.50265	-0.862606	6.634899
73	6.94463	-0.905792	5.164678
74	4.330822	-1.024311	4.922065
75	2.916625	-1.574503	4.736594
76	2.272161	-1.11322	4.227921
77	0.778222	-1.336082	4.539209

Prueba 4

	Pitch	Roll	Yaw
1	0.888318	-0.591747	3.19829
2	-1.402254	1.20518	-1.160346
3	-2.868693	-1.346828	3.284843
4	-3.395549	-1.410602	3.384947
5	-2.521105	0.615753	2.045724
6	-3.821086	0.421338	2.170857
7	-1.628801	-0.306572	0.987157
8	-2.955778	0.745123	2.005221
9	-2.67857	-0.030964	0.586493
10	-3.027182	0.502636	2.063323
11	-2.023982	0.278452	4.737489
12	-2.302012	0.638745	3.013225
13	-4.289098	-1.474769	2.824811
14	-4.989889	-1.269562	3.568045
15	-9.266427	-1.425108	3.457179
16	-11.27128	-2.043542	3.398415
17	-10.58249	-1.447552	1.514975
18	-11.21244	-0.752774	2.042568
19	-12.70385	-0.190186	3.951799
20	-11.81642	-2.096913	3.906825
21	-13.47111	-2.005252	2.762249
22	-11.92552	-1.785462	3.128506
23	-12.81424	-0.165966	2.497653
24	-15.26847	1.808399	-4.430355
25	-18.14475	3.459082	-4.157294
26	-18.55111	3.640832	-4.745517
27	-19.19064	5.535967	-3.30308
28	-18.28078	3.659365	-2.61429
29	-19.00796	2.467511	-4.020953
30	-18.8918	5.154651	-6.212991
31	-17.67233	6.656806	-7.534072
32	-17.06842	3.813455	-2.287045
33	-19.06732	4.405113	-2.615325
34	-18.47735	4.546788	-1.659141
35	-19.2933	5.224197	-3.015191
36	-17.05589	6.431139	-3.018944
37	-15.63509	2.648326	-3.122321
38	-13.75415	-0.502968	5.351049
39	-12.76406	-1.597964	5.03851
40	-13.12598	-2.140128	5.012906
41	-11.86974	-1.687086	4.020486
42	-11.50316	-2.349767	5.2182
43	-6.540056	-1.100009	4.671788
44	-4.172707	-1.359771	5.038783
45	-2.915591	-0.280059	2.854747
46	-1.410427	-0.766123	4.156964

	Pitch	Roll	Yaw
47	-1.37152	-0.86407	3.879279
48	-0.54605	-1.476188	4.273928
49	0.869415	-0.392005	7.074677
50	1.319902	-0.435328	7.880537
51	2.31636	-0.281402	7.640789
52	1.146569	-1.62647	4.678932
53	3.17147	-0.41187	4.409009
54	3.491318	-0.709383	4.723148
55	8.582525	-1.10562	6.93786
56	9.498733	-0.425192	4.610324
57	9.674508	-1.118784	5.400576
58	12.09585	0.004673	3.806817
59	12.11702	0.085667	3.950976
60	12.23774	-1.217225	4.707018
61	13.77119	-0.987974	6.65718
62	16.8706	-0.273468	5.331523
63	18.38581	0.317293	3.269342
64	17.45529	-1.369439	1.749317
65	17.22418	-1.979606	2.93143
66	15.89876	-1.180833	3.659335
67	17.02885	-0.462824	5.520551
68	18.08413	2.231216	-0.236102
69	17.38337	-0.417471	6.280426
70	18.74169	1.845861	-0.907638
71	17.11327	2.742088	-2.083864
72	17.96193	0.082453	2.853681
73	13.01314	-0.914113	5.404558
74	10.95301	-0.280345	5.168928
75	5.692129	-2.802932	5.420961
76	4.228104	-0.419923	3.663006
77	3.270886	-0.887653	2.941175
78	2.653048	-1.390176	3.576668
79	2.922917	-0.32893	6.609529
80	2.155117	-1.093696	4.084727
81	2.718108	-1.057287	4.96767
82	1.630694	-1.529536	4.641202
83	2.344325	-0.387375	6.605734
84	2.428843	-0.741674	7.488346
85	1.646071	-0.132633	5.77878
86	1.944107	-0.950079	3.513409
87	1.72347	-0.568032	3.231092
88	0.97212	-0.419153	2.171317
89	0.829379	-0.507352	3.107228
90	0.958408	-0.468458	2.311149
91	-0.255379	-0.954076	1.730503
92	0.554557	-0.461143	2.020214

	Pitch	Roll	Yaw
93	-0.143725	-0.467971	1.706869

Diseño de aplicación basada en Kinect para vehículo inteligente

Anexo II. Datos movimiento Yaw

Prueba 1

	Pitch	Roll	Yaw
1	-13.25213	0.129616	1.867399
2	-10.69031	0.343988	2.79198
3	-10.68587	0.525	-0.40162
4	-11.38957	-0.013099	0.154554
5	-13.02902	1.500386	1.625837
6	-14.20233	1.573802	2.656703
7	-14.27123	0.704162	1.4489
8	-13.69007	1.29094	1.715684
9	-14.50451	1.655983	1.841875
10	-15.04065	1.274892	2.98535
11	-13.35464	1.011075	2.419625
12	-14.15428	0.349632	2.403825
13	-13.69803	-4.829213	6.258685
14	-12.45118	3.52752	7.576365
15	-11.52004	3.326004	7.403955
16	-12.71553	0.197778	10.293956
17	-11.37217	0.855659	9.729287
18	-10.68639	0.211517	10.1994
19	-13.73547	3.293815	16.167526
20	-12.93821	3.041478	16.123463
21	-10.17189	3.241128	17.23535
22	-11.03995	3.633615	15.479642
23	-9.675913	1.212954	17.942806
24	-10.71868	0.674562	17.661987
25	-11.47213	0.709528	18.937078
26	-10.96699	0.385402	17.941359
27	-10.99175	2.3845	19.948261
28	-11.76668	-1.932619	17.505049
29	-11.0175	-0.528477	21.654486
30	-11.98501	0.290892	21.535374
31	-10.97253	1.720779	23.035141
32	-11.56982	2.650736	21.997808
33	-11.95798	2.089463	20.999424
34	-10.0953	2.13652	19.471058
35	-11.45025	1.604572	23.884508
36	-10.93602	1.620454	22.533834
37	-11.68584	-1.707385	23.133926
38	-10.89158	1.484751	23.441029
39	-11.81078	-0.447765	22.582954
40	-10.90987	0.962338	20.282223
41	-12.21897	-0.253692	20.225203
42	-12.47507	-0.027997	18.642242
43	-11.11374	-0.172961	18.530012
44	-12.30628	0.741322	17.556919
45	-11.34838	2.452476	13.700455
46	-10.29498	2.522021	14.013723

	Pitch	Roll	Yaw
47	-12.42137	1.201718	12.088926
48	-12.07502	0.681687	10.811197
49	-13.39859	1.238877	12.079056
50	-12.49291	0.904018	12.765634
51	-12.74064	0.244606	13.032073
52	-12.19224	1.09282	12.710961
53	-11.88942	1.328317	8.553727
54	-11.83464	1.321488	8.689488
55	-13.38258	2.229462	8.8137
56	-13.41606	2.933792	8.7693
57	-13.31498	2.796402	8.944743
58	-13.70922	1.912384	5.686836
59	-14.34679	2.197421	6.677303
60	-13.00791	-0.034691	2.009836
61	-13.94857	0.802998	1.938089
62	-12.6491	0.751012	3.497625
63	-16.00563	3.735478	-0.883619
64	-16.93813	3.455893	-1.362152
65	-14.28744	-0.508524	-3.284014
66	-13.92784	3.246827	-4.966672
67	-14.52582	3.834757	-5.088352
68	-14.59178	5.494507	-5.230307
69	-14.82522	3.740671	-7.031206
70	-15.17101	3.843633	-6.873428
71	-13.18789	1.341604	-12.357993
72	-12.57427	1.421864	-11.758375
73	-12.77961	-2.260666	-16.180937
74	-12.65741	0.111007	-13.922374
75	-13.57416	1.829967	-18.055515
76	-12.39364	1.756047	-18.900072
77	-11.87047	3.682283	-19.58577
78	-13.34739	0.591133	-20.278196
79	-13.22368	0.36329	-21.046989
80	-15.636	-0.995884	-22.846651
81	-16.40471	2.98789	-19.573448
82	-14.57554	-0.966525	-22.286861
83	-15.09689	-2.371062	-20.261868
84	-15.8115	-1.93183	-22.565069
85	-16.6195	4.132287	-20.553419
86	-16.14477	-0.813568	-23.18021
87	-14.95497	-2.191917	-21.311863
88	-13.13401	-5.054317	-24.479595
89	-17.61403	2.106069	-24.657921
90	-14.43966	0.527089	-26.431545
91	-17.35534	-2.668618	-23.419567
92	-16.49294	-2.745314	-22.194242

	Pitch	Roll	Yaw
93	-16.94427	-1.738008	-22.780926
94	-16.17691	-2.445077	-23.211241
95	-15.52524	-0.869628	-21.899427
96	-16.02888	-0.36611	-21.673283
97	-15.49275	-3.824055	-18.1266
98	-15.90491	-3.493001	-17.136433
99	-15.16912	-4.611049	-18.51899
100	-14.85772	-4.280106	-18.950346
101	-15.08774	-4.353438	-17.55912
102	-15.4182	-2.081105	-16.850779
103	-15.58347	-1.702353	-17.033459
104	-13.95342	-3.717048	-13.606995
105	-13.59008	-5.346892	-14.437531
106	-13.34752	-1.858008	-10.674369
107	-14.68208	-3.015802	-11.679813
108	-15.94039	-2.518857	-11.103369
109	-15.35003	-1.934333	-9.724001
110	-15.52635	-1.367964	-8.483441
111	-15.7075	-0.823063	-7.656369
112	-16.70606	-0.237328	-6.307721
113	-15.13306	0.22693	-4.730526
114	-15.34517	-1.153891	-5.238729
115	-16.44396	3.749766	-2.432904
116	-15.1997	2.22137	-1.264477
117	-16.38085	-0.23338	0.934815

Prueba 2

	Pitch	Roll	Yaw		Pitch	Roll	Yaw		Pitch	Roll	Yaw
1	-15.48599	-0.176914	-0.135336	47	-10.85616	-0.967038	21.328823	93	-14.48182	-0.739411	-2.854878
2	-15.07941	0.140646	0.352003	48	-10.50768	-0.561244	21.087448	94	-16.48591	-0.385093	1.9392
3	-14.38509	1.82039	2.817073	49	-11.71642	-0.975265	19.229767	95	-16.24278	-0.691307	-0.608938
4	-15.51548	0.142416	1.829159	50	-10.76215	-0.450817	21.832331	96	-14.18101	-1.365514	-2.718031
5	-15.1691	1.106175	2.70228	51	-11.50489	-0.343497	20.576191	97	-15.87768	-0.021564	-5.208755
6	-15.35156	-1.859057	-2.108738	52	-11.59697	-0.835722	21.10524	98	-14.39597	3.434705	-5.250087
7	-16.21285	-0.296212	1.764285	53	-12.00453	-0.834933	21.704952	99	-15.33229	2.802276	-4.52146
8	-13.37084	-0.821022	0.000365	54	-11.57528	1.060691	23.526142	100	-13.88354	-3.453339	-9.175669
9	-13.22205	-0.438867	1.429895	55	-11.58022	1.244695	24.107325	101	-12.42465	-3.430018	-9.356043
10	-14.49266	1.362088	4.694974	56	-11.97107	1.601462	24.755201	102	-13.8117	-2.939921	-9.633564
11	-15.90869	-0.678999	0.390998	57	-12.3499	-1.791728	22.997004	103	-14.91133	0.077336	-11.991574
12	-16.48735	0.305189	1.838309	58	-9.677024	1.741887	23.484913	104	-13.34081	-0.4686	-10.895895
13	-16.00459	0.920668	2.92316	59	-11.8803	-0.182455	24.34635	105	-14.74521	1.496911	-11.784256
14	-17.47673	0.444925	0.891187	60	-11.13784	3.282822	25.765806	106	-15.18348	-2.721279	-14.206223
15	-14.34377	-0.5131	4.174313	61	-11.59956	1.553302	23.476864	107	-14.77013	-1.523368	-13.675898
16	-15.59599	0.3749	6.287693	62	-12.62906	1.147155	23.990776	108	-13.46164	-0.213001	-17.361513
17	-14.23772	-1.1048	6.42568	63	-14.6535	0.490756	23.090322	109	-12.70123	1.437872	-16.969137
18	-14.56786	-0.469215	5.840964	64	-18.33131	-0.215051	21.619055	110	-14.67558	-0.738942	-20.382099
19	-14.92168	-0.354647	6.137819	65	-12.3616	-0.35351	21.947836	111	-15.55382	-1.82683	-19.570566
20	-13.94343	0.625072	7.166386	66	-12.34693	0.545903	24.120125	112	-15.74051	-1.150987	-19.723167
21	-14.98927	-1.117521	6.464752	67	-10.88898	-0.915773	21.784487	113	-16.56483	-1.973783	-19.600653
22	-12.76243	0.032967	7.993074	68	-11.86936	-1.781392	18.23839	114	-17.98786	-0.477747	-22.067179
23	-13.24347	-0.614487	5.631439	69	-12.73711	2.461473	15.481011	115	-18.36515	-0.859457	-23.334925
24	-13.21675	2.078774	9.849559	70	-11.34726	1.005397	15.82003	116	-17.26698	1.583301	-21.428587
25	-12.46851	1.017053	11.562428	71	-12.01424	0.472248	15.758431	117	-16.89226	2.175771	-20.997334
26	-13.6463	2.926006	11.697745	72	-12.85063	0.428445	15.662035	118	-15.56272	-1.690407	-24.777796
27	-14.45501	2.526228	10.09781	73	-12.86425	-0.347823	16.343048	119	-18.97331	-4.076861	-22.148874
28	-11.45722	0.588703	10.08045	74	-13.55003	1.212296	9.98482	120	-18.00765	-3.170023	-22.698477
29	-11.70499	0.672024	14.434612	75	-14.6451	1.898444	8.288681	121	-15.86847	-2.677563	-20.696541
30	-11.46447	0.588521	15.787587	76	-15.81821	2.114257	8.948803	122	-16.09677	-2.603825	-21.113852
31	-13.03837	2.233834	16.925821	77	-14.94778	1.639935	7.392039	123	-17.72539	0.767817	-16.434469
32	-11.17568	0.058104	14.91481	78	-16.11239	1.455155	7.949034	124	-15.27623	-0.575962	-21.989422
33	-10.3869	0.784637	16.415751	79	-16.86255	1.778797	8.414173	125	-16.65389	-1.770907	-17.686268
34	-11.94718	-0.150995	17.648817	80	-16.7372	3.428729	7.732456	126	-16.10591	-2.885252	-17.547382
35	-11.37945	1.450764	19.002111	81	-17.39577	3.123779	6.716294	127	-15.14519	-3.127977	-17.55657
36	-10.82934	1.166653	17.781923	82	-14.64215	1.020759	1.227644	128	-15.23615	-4.25679	-15.858932
37	-10.62163	1.314535	19.426956	83	-15.02334	-0.014871	0.620739	129	-15.42755	-3.287515	-16.546776
38	-11.50491	-1.10174	18.337968	84	-14.67457	-0.213967	0.318305	130	-15.56384	-2.78064	-17.884188
39	-12.59248	2.137727	18.384233	85	-15.40368	0.109896	0.631297	131	-15.25934	-2.475548	-13.976064
40	-12.28254	0.768212	20.954048	86	-15.15351	-0.140581	0.107466	132	-18.13409	-2.322271	-12.385604
41	-12.04553	0.553647	21.331587	87	-15.61215	0.487227	0.777674	133	-15.6463	-0.495198	-13.472176
42	-11.89337	0.38011	21.904016	88	-14.33656	-0.755157	0.761136	134	-15.55909	-0.702542	-12.860392
43	-11.01855	0.065887	22.356041	89	-15.00152	-0.354395	0.983037	135	-15.98574	0.989344	-6.966696
44	-10.61747	-0.562236	21.313986	90	-15.94088	-0.308168	0.798162	136	-16.01854	2.116799	-6.468919
45	-12.43239	0.15086	21.683475	91	-15.6392	0.25416	0.267152	137	-15.55044	-0.287986	-7.527857
46	-12.30177	-0.668684	21.467848	92	-15.43374	0.83625	2.375493	138	-15.22356	0.640051	-4.350404

	Pitch	Roll	Yaw
139	-16.11609	-0.56497	-3.211042
140	-15.46832	-0.658735	-3.70566
141	-16.65025	0.250758	-4.187204
142	-17.89861	-0.06189	-2.264353
143	-16.18837	-0.024971	-3.878605
144	-17.80686	0.513407	-1.690018
145	-13.34775	-3.806492	-1.709484
146	-14.3261	-3.652272	-2.633406
147	-12.97399	-4.61454	-2.253345

Prueba 3

	Pitch	Roll	Yaw
1	-16.16333	1.062534	1.781643
2	-17.48734	0.946578	-1.734948
3	-15.80193	2.96961	2.70281
4	-14.79239	-1.513873	3.815862
5	-17.71911	0.225326	5.414165
6	-17.39782	-0.072345	6.84135
7	-15.81924	1.426835	10.172623
8	-15.38069	1.378833	9.965777
9	-14.83295	2.771863	10.071829
10	-15.20112	3.046799	10.352962
11	-14.42705	2.88908	10.913801
12	-12.90294	0.771904	11.437841
13	-12.40233	1.048657	10.607829
14	-11.37964	1.045479	13.884291
15	-11.87889	-0.701191	12.382845
16	-11.8464	0.386149	12.525148
17	-13.36177	2.814641	14.764421
18	-12.06308	0.84933	15.571328
19	-11.26533	0.447132	17.077292
20	-12.44652	1.105497	17.644648
21	-11.28692	1.253466	18.633954
22	-13.17749	3.505997	16.08107
23	-11.2365	1.653449	16.086054
24	-12.97457	1.999828	18.589214
25	-11.41573	1.114369	20.951799
26	-13.47528	0.493421	19.549316
27	-11.70763	0.032481	19.555134
28	-12.51743	-0.057751	20.218239
29	-13.44204	1.607397	23.49852
30	-11.57653	0.69492	24.3316
31	-12.52198	-0.557488	20.088261
32	-10.42041	-0.532556	21.193455
33	-9.973671	0.315464	24.878143
34	-10.78669	0.53555	24.419134
35	-11.09579	1.719092	23.14859
36	-10.46489	1.752059	24.605047
37	-11.6702	1.369869	23.686115
38	-11.89927	-0.011299	20.547983
39	-11.75931	2.153258	23.943644
40	-15.94206	-0.744724	18.435314
41	-12.29699	1.820749	23.320229
42	-12.93822	0.016807	20.39179
43	-12.55332	-0.860188	20.61338
44	-12.44937	-0.3265	20.326242
45	-14.03294	-0.528278	20.107935
46	-13.51798	0.910547	22.037815

	Pitch	Roll	Yaw
47	-12.52661	-0.850247	19.809177
48	-12.43646	-0.398093	18.160429
49	-12.73853	1.58704	13.640591
50	-12.7836	-0.409327	14.443863
51	-14.08512	0.857143	14.767879
52	-13.82656	2.493915	18.120441
53	-12.9121	0.074193	13.015885
54	-12.3383	0.258885	14.069542
55	-11.57318	0.838311	14.991887
56	-12.22713	0.714416	16.155968
57	-13.41543	0.500235	9.042688
58	-13.31417	1.48286	7.586611
59	-14.63797	0.441133	4.35191
60	-15.15628	-0.043604	5.380593
61	-15.53622	-0.433556	5.466848
62	-14.55033	-0.869152	7.273904
63	-12.82647	0.325225	7.065948
64	-13.20801	-0.554576	5.863357
65	-13.94126	0.460867	4.923535
66	-14.4599	0.060019	5.108218
67	-12.95059	0.27935	4.432748
68	-14.26092	-0.487814	0.217631
69	-13.35178	0.526417	-0.865991
70	-13.40033	0.39827	-1.068468
71	-13.9531	0.620488	0.898077
72	-14.48626	1.315432	-0.301067
73	-12.47641	-0.147531	1.029581
74	-14.8287	0.020167	-0.048828
75	-8.743197	1.134638	1.310714
76	-9.141901	1.06601	2.311559
77	-8.448709	1.267209	3.520392
78	-8.814387	5.122331	-1.189268
79	-9.849689	3.930806	-0.122355
80	-9.111546	6.42644	-2.7751
81	-11.28425	2.524579	0.503591
82	-14.63892	-0.094215	-2.355409
83	-15.35075	2.819754	-2.102275
84	-12.99989	0.253323	-2.090823
85	-12.88997	0.447759	-5.514585
86	-13.87141	0.241732	-6.027532
87	-12.36757	-0.602247	-8.193001
88	-11.27711	-0.935683	-9.871236
89	-11.6888	-0.820079	-8.231098
90	-13.36704	-1.805967	-10.307218
91	-13.32923	-1.837148	-9.153761
92	-9.978382	3.853608	-8.872396

	Pitch	Roll	Yaw
93	-6.144369	2.590557	-16.665543
94	-3.761479	4.634789	-18.180643
95	-0.883524	6.433522	-20.108368
96	-4.553033	6.831439	-19.146343
97	-8.986505	1.957003	-18.906141
98	-9.119948	-3.213904	-20.846252
99	-8.240657	3.492772	-19.821072
100	-9.528003	0.6656	-17.527803
101	-11.01412	-3.237581	-19.917826
102	-12.28777	-5.585608	-18.643967
103	-12.26836	-0.554414	-20.440876
104	-10.95576	0.263333	-22.898745
105	-11.81218	-6.243963	-21.750603
106	-13.84354	-2.183291	-22.125437
107	-14.13301	-7.450958	-21.546661
108	-15.97746	0.360401	-22.294439
109	-19.54669	-8.007137	-19.690397
110	-17.01837	-2.752473	-18.766132
111	-16.47294	-3.431948	-21.759193
112	-16.2407	-3.757812	-22.076677
113	-15.44435	-5.852255	-22.690939
114	-15.08137	-6.650207	-22.885883
115	-15.29422	-6.642555	-21.688177
116	-17.26091	-5.775903	-20.517561
117	-15.42546	-1.386258	-22.740322
118	-12.97305	-6.722608	-18.664213
119	-15.32705	-2.402929	-13.284818
120	-11.77593	-0.915812	-10.678103
121	-11.85385	-1.846175	-10.613432
122	-12.40756	-1.232887	-7.448393
123	-15.14334	0.441431	-8.74146
124	-17.01984	1.312783	-4.242034
125	-17.58719	2.14921	-1.171387
126	-16.92117	1.928608	-0.997964
127	-18.38032	-0.126277	0.690484
128	-15.79932	0.729587	-2.742337
129	-15.5548	-0.127771	-2.203035
130	-18.38799	1.481142	4.035645
131	-18.96535	-2.862021	1.565781

Prueba 4

	Pitch	Roll	Yaw
1	-14.43465	1.292161	1.088549
2	-5.000144	-0.844688	0.482036
3	-12.48613	1.837533	0.332831
4	-10.85258	1.02891	-1.413592
5	-10.78747	2.228885	-2.012277
6	-11.28888	0.832317	-1.601215
7	-12.54273	-0.230527	-0.994416
8	-5.920717	0.702757	2.951023
9	-7.346068	3.042121	2.357282
10	-9.558924	-0.92768	4.481878
11	-9.94774	1.422525	11.526878
12	-8.54433	0.616142	10.143488
13	-10.07173	1.535164	13.308784
14	-7.156487	0.086829	13.665152
15	-8.82232	0.010463	16.346197
16	-8.499342	2.08966	16.82045
17	-8.946937	1.062645	17.481792
18	-9.411311	0.923599	19.259592
19	-8.637564	0.281689	18.383966
20	-6.672613	0.836406	22.021969
21	-6.565157	0.717947	21.869137
22	-7.868575	-0.764785	24.898394
23	-7.494996	-0.760298	25.652124
24	-6.022853	-5.631404	25.884136
25	-7.053359	-0.774392	24.747086
26	-7.033977	-2.567399	26.063978
27	-6.900841	-2.706028	26.303164
28	-7.287026	-0.611993	25.415878
29	-7.264508	-1.259179	26.003077
30	-6.076488	-0.991121	26.194426
31	-5.584529	-1.214405	25.57103
32	-7.010331	0.092002	24.557936
33	-5.897868	-0.671141	25.940374
34	-6.018632	-1.349802	26.858519
35	-6.606651	-0.602659	26.918888
36	-5.681955	-0.301155	25.649727
37	-7.005144	-0.861264	25.943769
38	-6.898702	0.572576	25.909582
39	-5.907917	-0.663024	27.956305
40	-6.513148	-0.261263	25.731451
41	-7.616522	0.262974	25.5776
42	-6.741948	0.063917	27.095896
43	-7.683743	0.770623	25.489422
44	-8.842443	1.448055	27.335945
45	-6.598439	-0.737359	30.041288
46	-7.523376	-0.531406	25.444553

	Pitch	Roll	Yaw
47	-7.293028	2.395937	26.534849
48	-7.59811	0.232428	25.439014
49	-8.46861	-2.292885	21.683435
50	-8.795151	-1.508663	22.742878
51	-8.770728	-0.327305	21.36838
52	-10.12773	-0.475863	19.525789
53	-9.166162	1.032526	17.622606
54	-9.527052	0.73202	18.769995
55	-7.968349	1.953943	16.596714
56	-9.511297	1.959547	15.985805
57	-11.00288	1.753317	16.170015
58	-9.952132	1.118305	12.66257
59	-9.782681	0.533151	13.893485
60	-9.892198	-0.461289	13.551881
61	-13.32035	1.034872	12.918478
62	-12.24912	0.396989	10.557793
63	-11.79905	-0.466956	10.56523
64	-10.19693	3.3045	10.768133
65	-14.75788	1.980171	5.790114
66	-10.08238	1.086547	4.661455
67	-10.46482	0.173126	-1.483765
68	-11.53111	-0.95808	-5.628142
69	-11.0954	0.289365	-6.290806
70	-11.58788	0.435465	-5.728278
71	-9.167822	-0.131928	-9.305748
72	-9.859878	-0.783312	-8.511735
73	-10.2454	-0.522619	-8.556597
74	-10.28698	-0.387764	-11.643631
75	-9.307427	-1.950932	-14.368645
76	-17.2589	1.839677	-13.480711
77	-19.52888	-0.037198	-14.830903
78	-18.94587	1.496424	-14.160738
79	-15.0104	-3.241509	-17.319525
80	-9.960954	-2.826433	-19.597773
81	-18.32511	-2.811237	-19.072639
82	-18.69306	-1.550931	-18.123964
83	-21.33091	0.646615	-21.519176
84	-15.33821	0.111109	-20.067965
85	-13.69629	-1.607801	-20.907955
86	-14.29618	-0.097137	-21.506573
87	-14.81433	2.084873	-19.734177
88	-17.51842	-0.026605	-20.745831
89	-19.15731	-1.427854	-21.781536
90	-16.70843	-3.790994	-20.320194
91	-13.03784	-4.350227	-22.582407
92	-13.28464	-4.014622	-21.971626

	Pitch	Roll	Yaw
93	-15.48832	-4.801761	-20.729672
94	-12.81537	-3.135325	-19.773165
95	-12.0416	-2.684786	-21.744236
96	-15.88257	-6.557398	-20.721353
97	-17.42991	-9.518466	-21.401856
98	-15.84859	-2.023062	-20.510597
99	-10.5208	0.84513	-19.812759
100	-15.99414	-1.634557	-19.218319
101	-11.24895	-3.427796	-21.287865
102	-12.29839	-5.816652	-20.004011
103	-15.59513	-3.298083	-21.081074
104	-17.36415	-4.727029	-18.402996
105	-14.5715	-7.434062	-18.977662
106	-15.1254	-3.296286	-17.615904
107	-13.66951	-1.012006	-16.237555
108	-12.6497	-0.198863	-16.690968
109	-21.4266	-0.909527	-15.776899
110	-16.45402	-0.663861	-12.003616
111	-17.78023	-1.079725	-12.520695
112	-19.5664	-7.153537	-8.459249
113	-15.86393	-4.360096	-9.167337
114	-16.58406	-6.257985	-8.534022
115	-14.01733	-5.107859	-5.80251
116	-16.0026	-4.233721	-7.370452
117	-16.87469	-1.869929	-6.91341
118	-12.38956	-2.896897	-4.874168
119	-11.03109	-7.50453	-2.197865
120	-10.5231	-1.1547	0.888756
121	-11.57051	-5.677444	-1.658543
122	-18.55405	0.909136	-3.454769
123	-18.67807	2.311121	-2.807562
124	-20.53342	1.675691	-1.669373
125	-21.60801	1.041818	-1.045135
126	-20.65731	1.693928	-1.423083
127	-21.25069	2.850468	-0.255759
128	-21.38242	0.4364	-1.360499

Diseño de aplicación basada en Kinect para vehículo inteligente

Anexo III. Datos movimiento Roll

Prueba 1

	Pitch	Roll	Yaw
1	-3.542662	0.093282	-3.236325
2	-3.667994	-1.504003	-1.57542
3	-9.653043	-1.040303	2.074647
4	-9.802851	-0.398875	0.15871
5	-8.78559	-0.8621	0.914463
6	-8.69197	-0.586923	1.596276
7	-10.04083	-1.120926	1.719995
8	-7.626934	-0.952622	3.172445
9	-8.956435	-1.310809	2.109461
10	-7.462265	-1.402936	3.444448
11	-7.715395	-1.078928	3.815509
12	-8.911525	-4.780056	3.342822
13	-8.426103	-3.049256	2.327173
14	-9.089281	-6.583703	2.356992
15	-8.518894	-7.158992	1.065272
16	-9.039381	-8.106551	0.467325
17	-9.569856	-10.54755	1.488411
18	-10.62654	-12.21606	2.359496
19	-8.862509	-13.49183	7.175764
20	-7.793978	-15.31784	0.73201
21	-9.993229	-14.50436	4.807712
22	-12.10766	-16.20793	3.150465
23	-12.32437	-15.48405	-0.395991
24	-10.76145	-17.84869	-5.034742
25	-11.60659	-17.18572	-1.321028
26	-11.92749	-18.28998	2.434538
27	-12.20187	-17.23525	-0.739091
28	-12.42359	-15.94341	1.413086
29	-12.79728	-14.53928	-0.433434
30	-12.12997	-12.39886	-2.995778
31	-10.49315	-10.16796	-1.962099
32	-9.645056	-8.675692	1.416051
33	-10.03263	-9.617654	0.702514
34	-7.660666	-4.944931	-0.045776
35	-7.63199	-4.62633	0.739097
36	-9.366979	-5.915883	0.05403
37	-9.396568	-1.001272	-2.324803
38	-8.518303	-2.44736	-4.139213
39	-7.751224	-1.949906	-4.348654
40	-7.622804	-1.225409	-2.961424
41	-7.495922	-1.662876	-2.10623
42	-7.411747	-1.442035	-0.347337
43	-8.108387	-0.368716	1.148205
44	-8.104427	-0.142841	0.999598
45	-8.392384	-0.802082	0.731498
46	-8.147788	0.123343	0.643987
47	-8.191193	-0.305041	0.015879
48	-7.69	0.775137	2.315879
49	-8.031637	0.620536	1.196807
50	-8.471739	0.339713	1.770651
51	-9.203919	1.020846	0.355417
52	-8.362024	0.328606	2.437148
53	-8.969788	2.066088	-2.247003
54	-8.79643	2.018099	3.251665
55	-8.648461	2.668772	5.025528
56	-8.733701	1.943463	3.915028
57	-8.540392	1.948448	3.656245
58	-8.036489	1.84848	3.07683
59	-8.301788	4.880855	-0.909473
60	-8.932755	5.797992	-1.163378
61	-7.290396	4.836057	-2.182705
62	-8.707542	12.543363	-3.406064
63	-8.989582	12.402927	3.510921
64	-7.986122	13.705954	-3.053467
65	-6.524443	13.83275	-6.728733
66	-6.42729	14.193968	-5.623388
67	-8.244155	13.966926	-3.364842
68	-9.031028	14.578081	-0.909358
69	-10.40503	14.238909	-4.364844
70	-9.000905	15.023855	-6.318777
71	-9.947746	15.877218	-4.874196
72	-10.94339	15.463331	-4.858742
73	-10.38795	16.64187	-3.597358
74	-9.486959	17.027681	-6.413305
75	-10.02821	17.0741	-5.547853
76	-9.794705	18.46064	-6.214143
77	-9.451486	16.732439	-5.393841
78	-9.967717	17.176374	-6.492202
79	-10.59399	15.467789	-4.594197
80	-8.543511	15.974077	-7.025432
81	-9.09724	16.650795	-6.16676
82	-8.753652	16.375919	-10.31338
83	-10.32504	16.154745	-4.542032
84	-9.918951	14.063597	-0.845331
85	-9.398361	11.574054	0.375536
86	-10.88858	12.720591	0.599605
87	-10.08188	11.186352	1.986889
88	-10.05245	9.852331	1.052326
89	-10.51429	10.447172	0.714763
90	-9.993013	8.376365	2.435215
91	-9.537608	9.138275	-0.504051
92	-10.19615	7.718155	0.130241
93	-9.623293	9.398513	-0.55063
94	-10.4681	6.542255	4.391743
95	-10.00601	4.111622	4.114358
96	-10.3252	3.700906	4.263313
97	-9.588465	2.753806	3.752211
98	-9.388891	2.58835	3.077958
99	-9.340637	1.876774	2.151364
100	-9.498136	1.914822	1.44514
101	-9.591161	0.969131	5.588786
102	-9.327184	0.034755	5.342827
103	-8.691726	-0.392065	5.07769
104	-3.959758	-1.611083	-1.77942
105	-6.392586	0.188255	-2.39733
106	-7.239404	-0.632218	3.757809
107	-9.088088	-1.123764	3.284142
108	-9.309866	-1.017403	2.950035
109	-9.203044	-0.709044	2.255272

Prueba 2

	Pitch	Roll	Yaw
1	-9.176584	-0.528897	4.047364
2	-11.08154	-1.950325	2.653678
3	-10.21888	-0.647236	2.628257
4	-9.455641	-0.929171	2.582855
5	-8.677835	-1.647052	-0.165585
6	-8.649168	-1.463732	-0.104243
7	-8.100459	-2.459074	0.782262
8	-9.054725	-0.859573	-1.85642
9	-8.696433	-1.619009	-1.233009
10	-8.434516	-2.240239	-3.81485
11	-10.3734	1.282661	-7.850959
12	-11.76468	-3.277793	-8.700504
13	-11.36113	-1.840654	-9.597846
14	-11.58763	-1.721151	-9.81374
15	-10.67853	-4.513092	-0.927238
16	-9.69347	-3.391833	0.725995
17	-13.43717	-5.539508	-6.610732
18	-12.85954	-7.410997	4.246061
19	-11.70665	-9.729184	-1.768835
20	-12.72609	-11.159	-1.171004
21	-11.52715	-10.65905	-1.093948
22	-11.27882	-10.47648	0.924798
23	-11.7528	-13.47422	9.881198
24	-10.90811	-12.41201	7.819279
25	-12.65528	-13.95375	1.297246
26	-10.5575	-15.61327	4.912505
27	-12.12608	-15.48789	2.949021
28	-11.32949	-16.03986	6.833182
29	-11.06534	-15.61569	-1.968064
30	-11.76652	-15.16931	-2.386702
31	-12.94408	-16.07993	-7.863304
32	-10.00783	-17.28829	6.806784
33	-9.402439	-15.12374	6.013456
34	-11.53942	-16.91876	-2.65818
35	-12.18814	-16.27583	3.101246
36	-11.93026	-18.19898	8.149337
37	-11.52868	-17.78208	-3.006244
38	-14.37022	-17.16875	-5.015141
39	-12.72243	-18.17894	-4.380138
40	-12.09927	-11.75941	-0.433586
41	-12.26946	-9.565119	-6.394821
42	-11.25618	-10.36028	0.027932
43	-13.26947	-7.409796	-4.734193
44	-13.50502	-7.386234	-3.055394
45	-11.17514	-7.574404	-8.843332
46	-14.21753	-6.470738	-4.835243
47	-14.1515	-7.407744	3.289406
48	-12.09136	-5.67343	-3.56918
49	-12.02169	-3.930523	-6.729443
50	-10.49463	-4.976041	-5.823109
51	-10.02283	-2.990004	1.008664
52	-9.948677	-2.35647	4.12536
53	-8.44018	-0.1732	0.058499
54	-10.22598	-0.011594	-1.038522
55	-10.66899	0.320926	0.363977
56	-9.798374	-0.28359	1.033074
57	-9.877874	1.325526	2.33563
58	-9.03347	0.607149	2.172059
59	-8.961706	0.255284	1.48415
60	-9.281377	0.602144	2.615524
61	-9.255588	1.513549	0.502819
62	-11.2231	-0.243765	-0.046
63	-11.1563	3.361376	-3.493757
64	-10.23319	1.7346	2.475844
65	-10.79951	2.121875	-2.574587
66	-11.12713	2.988134	-1.730169
67	-8.966934	3.680994	-2.012149
68	-9.505002	3.953938	-1.74797
69	-10.27272	6.290004	0.016751
70	-10.77167	6.716893	1.109944
71	-10.09022	5.521501	-5.990516
72	-10.01207	8.425212	-5.569026
73	-9.861744	9.141389	1.349266
74	-9.79529	10.79176	-5.160014
75	-10.07931	9.85871	-3.208133
76	-10.17289	10.837947	-6.529138
77	-9.933017	10.462707	-5.771119
78	-7.81292	11.809495	-8.047565
79	-8.293893	12.165833	-8.029557
80	-9.086172	14.082726	-6.794337
81	-8.965755	14.733057	-7.734393
82	-7.891657	13.340947	-13.44703
83	-7.816767	15.657716	-11.15103
84	-9.112099	15.717152	-9.452009
85	-6.604201	15.84466	-10.58697
86	-8.305854	17.367023	-11.00097
87	-7.206993	16.507317	-11.78071
88	-7.417289	16.507534	-10.30113
89	-6.118375	15.123742	-9.590417
90	-7.4653	16.48033	-10.7877
91	-7.645038	15.115999	-10.08981
92	-8.531121	15.980591	-6.149341
93	-6.95746	16.435446	-6.89732
94	-7.886742	16.069159	-9.7622
95	-9.553388	12.906372	-4.4571
96	-9.554193	11.217936	-6.99602
97	-10.0834	10.444269	-2.72001
98	-10.71361	11.264977	-6.2485
99	-9.597638	10.266982	-3.29714
100	-10.34222	8.865329	-3.26687
101	-10.06042	9.397662	-2.46794
102	-10.90705	9.224361	-4.30081
103	-11.9453	5.836208	-4.90381
104	-10.63493	6.527456	0.064572
105	-10.9621	7.749109	-3.70866
106	-11.88872	6.251675	-5.97522
107	-10.55055	5.679928	-2.38357
108	-11.85938	2.285759	-1.77002
109	-11.60998	1.944544	-2.61273
110	-10.6107	1.952372	3.054542
111	-10.16232	0.676733	1.403423
112	-11.69445	2.267372	4.267081
113	-10.99563	0.334571	-0.0681
114	-9.405536	0.253292	-0.38028
115	-9.021032	-0.742792	1.056404
116	-9.574593	-0.419012	0.629824
117	-9.323147	-0.513931	0.693622
118	-9.516972	-1.129439	1.098147

Prueba 3

	Pitch	Roll	Yaw
1	-11.39785	1.048381	-0.836974
2	-11.88423	0.301033	-0.663979
3	-11.74393	1.285165	-1.793212
4	-13.11803	2.756638	-1.182818
5	-12.0311	-0.364395	4.090877
6	-11.91736	1.024818	-2.62892
7	-12.26679	1.68496	0.995322
8	-11.66074	-0.126921	3.619338
9	-9.748677	-0.281936	1.900439
10	-9.364913	-0.093675	2.202175
11	-9.694714	-0.531026	1.351491
12	-10.21861	-0.611364	-0.209093
13	-9.568155	-1.045905	-1.218002
14	-9.62031	-1.123103	-3.132003
15	-9.396425	-1.141544	-2.999674
16	-8.803143	-3.649481	-2.262326
17	-10.85598	-5.856991	-2.251043
18	-11.91125	-5.349705	-3.29592
19	-11.44408	-7.394899	-4.065558
20	-10.14661	-8.079153	0.736365
21	-13.13257	-7.565926	-2.038653
22	-12.27786	-9.785007	-1.804749
23	-12.09127	-10.8559	0.312552
24	-10.8807	-12.57034	0.230909
25	-14.03289	-14.77604	4.371187
26	-11.98704	-14.47797	4.128209
27	-13.10803	-16.73446	4.22433
28	-13.75861	-15.26085	-1.172043
29	-11.67477	-16.87285	-1.267802
30	-14.03626	-17.90556	-10.71143
31	-14.89184	-16.48288	-4.358283
32	-14.7199	-15.55765	-4.918735
33	-11.18973	-15.28724	1.822342
34	-11.31392	-16.99906	-11.24038
35	-12.5813	-16.30752	-5.399101
36	-10.35507	-15.30435	2.370266
37	-9.207858	-13.77372	3.698766
38	-9.453874	-11.80521	6.058167
39	-8.331266	-12.62566	4.025081
40	-9.541021	-10.70541	5.424328
41	-9.192247	-8.617088	5.478713
42	-9.509023	-6.768139	5.022372
43	-9.706685	-7.272888	2.204268
44	-9.52459	-3.306222	1.927045
45	-9.423519	0.705433	8.054906
46	-9.496318	0.441265	8.757341
47	-9.403952	0.370835	2.196762
48	-9.340125	0.866892	0.098714
49	-9.677178	1.744913	1.537408
50	-10.11977	2.092093	1.514214
51	-9.759224	1.571144	1.558752
52	-9.603277	1.464487	2.768638
53	-9.307033	1.47298	2.121987
54	-10.12063	1.167591	-2.274755
55	-12.00449	1.219812	-0.260466
56	-13.02013	3.426759	-3.681648
57	-12.75256	3.131732	-3.05786
58	-12.41439	2.842657	-3.750212
59	-12.33942	2.799498	-3.350311
60	-13.20178	9.214852	-6.057327
61	-11.70308	12.373206	-4.163131
62	-13.40596	12.002895	-6.286918
63	-12.48421	13.12989	-4.84887
64	-11.98783	14.585238	-4.830152
65	-12.13426	14.371984	-5.901415
66	-11.7858	13.827882	-6.723793
67	-11.83592	14.395539	-6.06885
68	-9.929673	15.390515	-6.612529
69	-10.5977	16.876339	-5.550991
70	-10.71911	16.881704	-6.063406
71	-9.875665	17.464541	-8.426475
72	-9.149683	15.408992	-12.32701
73	-10.30165	16.967623	-9.923142
74	-9.729225	16.336319	-10.16182
75	-8.096135	16.494083	-12.59073
76	-9.619866	15.787468	-10.53298
77	-9.972431	20.132231	-10.35617
78	-8.303373	19.60006	-13.54775
79	-8.673003	20.421989	-13.07528
80	-9.297105	20.291332	-11.89019
81	-8.338998	19.796556	-14.64985
82	-10.04535	17.301725	-10.82693
83	-12.5835	19.676142	-6.003704
84	-12.33578	14.062669	-5.595314
85	-13.37113	6.883294	-5.444128
86	-11.96559	1.936905	-0.051143
87	-12.78854	2.466277	-1.979797
88	-12.59349	3.595416	0.986927
89	-12.80688	2.133975	-3.838273
90	-11.81321	0.779035	0.892769
91	-12.20094	-0.045257	0.915659
92	-9.79692	1.053013	0.670297
93	-10.75859	0.708437	-1.085
94	-11.39579	1.811345	0.860316
95	-11.37791	1.524533	-0.04699
96	-11.30751	1.285072	-1.14193
97	-9.109821	1.793532	-0.42538
98	-11.49942	2.072013	-1.05415
99	-12.2805	0.91025	5.934586
100	-13.37112	1.134946	-1.23226
101	-9.881896	1.286771	5.749524
102	-9.975145	0.167214	3.102847
103	-9.896714	0.266696	1.302972
104	-9.342457	0.15349	0.835312
105	-9.849349	0.347933	0.885569

Prueba 4

	Pitch	Roll	Yaw
1	-9.804329	-0.214163	-2.739475
2	-11.27765	0.674342	-3.557695
3	-10.95146	-0.167294	-4.130698
4	-10.36773	-0.405025	-3.993409
5	-11.28645	0.317902	-5.435216
6	-9.628482	-1.549831	3.950327
7	-10.25834	-4.138668	-4.117712
8	-10.57854	-1.031734	-1.247201
9	-10.61222	-2.127916	-0.829552
10	-11.34644	-1.352591	-1.996441
11	-11.4261	-0.520725	-1.835331
12	-11.32227	-3.094382	-0.967511
13	-11.45731	-2.691255	-0.672489
14	-11.24226	-2.62841	-2.956164
15	-11.56259	-2.901165	-3.075189
16	-13.09582	-2.182067	-8.607442
17	-11.50873	-2.755558	-3.964132
18	-13.26785	-8.299511	7.004204
19	-12.63608	-12.06065	-8.448866
20	-12.27375	-11.09173	-8.597089
21	-12.35873	-14.08561	-9.079967
22	-13.73645	-14.18314	2.70039
23	-12.55875	-16.19412	2.912338
24	-12.68491	-16.77059	-13.6135
25	-9.254248	-15.62462	5.492784
26	-11.48564	-16.60594	5.993401
27	-11.93208	-15.06909	6.176659
28	-11.51308	-16.28108	0.600342
29	-10.21214	-14.42846	7.931887
30	-9.027802	-15.59908	-3.017294
31	-13.29103	-16.0035	-9.14396
32	-9.46785	-16.55967	-4.559069
33	-13.24309	-17.08776	-4.089327
34	-10.63695	-17.54599	4.136726
35	-10.2084	-15.72593	-6.566294
36	-8.849873	-16.46474	-9.421384
37	-9.739209	-14.5585	-9.400075
38	-11.59294	-16.23889	3.10412
39	-13.86385	-14.62019	3.947048
40	-12.63727	-14.66353	7.374645
41	-13.7034	-15.78874	6.820815
42	-12.91515	-16.21257	8.353855
43	-13.94191	-14.82003	5.5971
44	-11.34826	-15.61955	6.4947
45	-13.80062	-18.14928	7.619
46	-13.22496	-16.47455	9.350745

	Pitch	Roll	Yaw
47	-10.67689	-17.09786	9.921587
48	-12.2118	-12.78274	8.321798
49	-11.19798	-8.8518	3.552506
50	-10.24918	-3.913869	0.797207
51	-9.51138	0.307624	3.900376
52	-11.0912	-0.109366	3.161602
53	-12.11156	-0.182561	3.958559
54	-10.73262	0.428503	-0.215932
55	-10.13818	1.15862	2.38049
56	-10.36806	1.347202	3.94159
57	-10.39168	0.646634	1.105954
58	-9.821465	1.131189	2.750017
59	-9.777336	0.963316	2.011265
60	-9.964589	1.30889	2.018319
61	-9.789279	1.327941	2.153222
62	-9.499359	1.051734	2.909619
63	-10.45662	1.378183	3.177803
64	-9.925832	0.897299	6.83936
65	-10.04245	0.529154	7.222822
66	-9.865965	2.473975	1.969174
67	-9.593451	3.543506	6.50173
68	-10.79844	2.981629	5.074525
69	-10.6269	4.339745	0.122193
70	-11.2267	7.853325	1.470827
71	-9.871432	8.027279	2.012068
72	-10.08135	11.533572	-1.694978
73	-8.84904	10.191626	-2.896353
74	-9.794951	11.900806	-1.68801
75	-9.16245	15.792383	-1.78869
76	-9.628865	16.257271	-6.435743
77	-9.305055	15.077155	-1.779676
78	-10.6464	14.638171	-1.120002
79	-9.856895	15.944292	-2.987202
80	-10.9955	17.483332	-4.196177
81	-9.440095	14.536758	-8.226009
82	-10.07056	17.325771	-7.30752
83	-8.916913	16.478876	-7.265317
84	-9.997033	18.506124	-7.086235
85	-9.843701	17.316513	-4.60796
86	-10.96879	17.799726	1.836948
87	-8.126804	13.106235	1.798494
88	-10.30195	11.929234	-1.848248
89	-10.33374	10.591742	0.863819
90	-10.26866	12.806612	-1.962175
91	-10.26544	8.746322	5.039328
92	-10.20029	10.149266	1.326977

	Pitch	Roll	Yaw
93	-10.4005	6.997527	8.535892
94	-8.947847	8.463456	7.549687
95	-9.375123	8.190143	4.207486
96	-10.3745	2.069036	5.940098
97	-8.74738	-0.381521	3.005298
98	-10.55894	2.471215	4.550517
99	-10.98299	0.555728	10.05921
100	-9.963293	0.753956	0.422524
101	-9.872071	0.61833	1.200811
102	-10.7243	0.18748	-0.78691
103	-11.22409	-0.780214	1.031058
104	-11.32939	-2.329704	7.794286
105	-11.42219	-1.000808	6.798164

Diseño de aplicación basada en Kinect para vehículo inteligente

Anexo IV. Datos falso positivo

Falso positivo

	Pitch	Roll	Yaw
1	0.701306	0.052651	-1.726145
2	-0.163231	-0.192944	-0.315762
3	-2.798476	0.020157	-3.034225
4	-0.287903	0.484428	-2.743783
5	-3.611408	-0.360001	-1.309944
6	-0.123445	0.526328	-2.843144
7	-1.639669	1.094158	3.098816
8	-0.975375	0.885412	5.192731
9	-1.715243	1.175818	5.296705
10	-1.720633	1.707032	8.555183
11	-0.952464	2.021937	7.555521
12	-10.82618	6.811981	10.78723
13	-16.09874	1.975543	11.497974
14	-17.95545	10.67583	13.49492
15	-18.51871	11.092558	14.699948
16	-17.98078	9.75277	14.003886
17	-20.81899	2.167994	18.928975
18	-14.09481	4.094852	21.626673
19	-11.37421	0.519627	23.760739
20	-8.407327	-0.187728	24.787148
21	-6.600865	1.871469	23.092297
22	-4.593163	-4.034064	22.988884
23	-2.761417	-6.261777	24.790282
24	3.140892	-1.389109	22.22079
25	6.61148	-1.757191	23.222288
26	7.629158	-1.614282	23.733999
27	11.839695	-0.572939	24.054882
28	13.376811	-4.317801	24.790473
29	16.944361	-2.891576	-21.1296
30	19.668789	-2.336626	-23.69071
31	22.373156	-1.272888	-22.38826
32	30.308325	-3.041022	24.630055
33	-10.97838	1.02583	23.053632
34	-17.23181	-3.392303	21.378571
35	-17.72029	-0.684525	16.428836
36	-10.50507	-0.951681	17.196625
37	-4.296006	2.206668	14.879187
38	-4.574223	5.578358	13.118599
39	-8.037986	5.261584	8.652149
40	-9.855627	0.172408	9.786048
41	-9.842628	-0.964089	6.709383
42	-9.535663	-0.795869	2.780866
43	-8.734225	0.12421	-1.849092
44	-9.047172	0.232366	-2.191544
45	-8.432401	1.482833	1.411066
46	-7.707931	-2.024907	0.250957
47	-9.324106	0.16494	-0.628501
48	-8.237886	3.864262	2.191115
49	-12.31483	1.985465	1.663797
50	-14.5057	0.991082	2.365529
51	-14.04137	-0.590237	-1.31557
52	-14.65432	1.437067	-1.449863
53	-8.150301	-1.757459	-4.92091
54	-7.126181	7.453195	-4.238349
55	-8.314453	-1.249256	-6.535969
56	-7.069463	3.815951	-9.997845
57	-9.245545	7.119615	-11.76512
58	-9.773897	9.900202	-13.81306
59	-12.73497	13.424017	-17.26488
60	-14.28958	17.158787	-18.69597
61	-5.695352	24.356604	-19.93842
62	-2.506476	8.526159	-19.21571
63	-1.38684	9.281582	-21.86314
64	-3.495826	6.767931	-23.91198
65	-1.02169	11.473749	-25.20092
66	-1.235613	11.058143	-24.08576
67	-1.148739	9.379582	-24.82988
68	2.603392	15.310476	-23.81702
69	-1.628294	9.968738	-23.57729
70	-0.696924	10.319089	-24.61187
71	-1.586241	8.683735	-23.90724
72	-1.233526	8.390989	-23.38171
73	-0.279075	9.165975	22.028953
74	-1.909309	8.203281	25.519515
75	-1.10937	8.127612	25.081667
76	-0.484039	8.626097	23.972558
77	109.3579	-110.5633	-23.16273
78	5.588781	19.43117	-23.57133
79	3.532579	14.43628	-21.07099
80	4.718618	16.219791	-18.15913
81	4.527471	16.842285	-17.11456
82	4.140788	15.370608	-13.23763
83	4.604634	15.541183	-14.01613
84	7.742913	17.661077	-12.19415
85	7.605412	12.598895	-11.15956
86	7.153679	13.214535	-9.013283
87	6.907511	16.172409	-9.299091
88	3.32884	15.174366	-7.755489
89	2.351277	16.015261	-4.250971
90	2.192887	19.209986	-5.412321
91	3.696169	17.262186	-5.62149
92	5.876042	7.729407	-3.493566
93	5.833568	11.884578	-2.02148